First Hit Fwd Refs

Generate Collection Print

L6: Entry 4 of 62 File: USPT Feb 17, 2004

US-PAT-NO: 6694381
DOCUMENT-IDENTIFIER: US 6694381 B1

TITLE: Platform-independent communications protocol supporting communications between a processor and subsystem controller

DATE-ISSUED: February 17, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Lo; Horatio	Milpitas	CA		
Lee; David	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Vicom Systems, Inc.	Fremont	CA			02

APPL-NO: 09/ 523027 [PALM]
DATE FILED: March 10, 2000

PARENT-CASE:
This application is a continuation of Ser. No. 08/992, 202 filed Dec. 17, 1997 now U.S. Pat. No. 6,078,968.

INT-CL: [07] G06 F 3/00

US-CL-ISSUED: 710/5; 710/3, 710/4, 710/11, 710/30
US-CL-CURRENT: 710/5; 710/11, 710/3, 710/30, 710/4

FIELD-OF-SEARCH: 710/5, 710/11, 710/3, 710/4, 710/30

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4807121	February 1989	Halford	364/200
5363484	November 1994	Desnoyers et al.	395/200
5418960	May 1995	Munroe	395/700
5421014	May 1995	Bucher	395/650

☐	5724599	March 1998	Balmer et al.	395/800
☐	6038400	March 2000	Bell et al.	710/11

ART-UNIT: 2182

PRIMARY-EXAMINER: Huynh; Kim

ASSISTANT-EXAMINER: Kim; Harold

ATTY-AGENT-FIRM: Campbell Stephenson Ascolese LLP Campbell, III; Samuel G.

ABSTRACT:

A method and apparatus implement a communications protocol whereby a host application program can communicate with a computer subsystem without the use of special driver software. In this way, the application program is able to invoke virtually any vendor-unique function on a compatible subsystem controller using only standard read/write system calls. This avoids platform dependency and greatly improves the portability of the application program.

55 Claims, 6 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L6: Entry 4 of 62 File: USPT Feb 17, 2004

DOCUMENT-IDENTIFIER: US 6694381 B1
TITLE: Platform-independent communications protocol supporting communications between a processor and subsystem controller

Brief Summary Text (2):
The present invention relates to computer subsystems. More particularly, the present invention relates to a method and apparatus for communicating with one or more computer subsystems via a platform-independent communication protocol.

Brief Summary Text (8):
The present invention provides techniques that allow a host application to communicate with one or more computer subsystems via a communication protocol that is platform independent.

Drawing Description Text (3):
FIG. 2A depicts a platform-independent subsystem protocol in accordance with one embodiment of the present invention;

Drawing Description Text (6):
FIG. 3 depicts the steps of performing a write operation conforming to platform-independent subsystem protocol in accordance with one embodiment of the present invention; and

Drawing Description Text (7):
FIG. 4 depicts the steps of performing a read operation conforming to platform-independent subsystem protocol in accordance with one embodiment of the present invention.

Detailed Description Text (3):
The present invention supports communication with a computer subsystem (e.g., a hard disk subsystem) by allowing the transfer of information using a communication protocol that obviates the need for a software driver to communicate with the subsystem. Application programs and subsystems employing the method of the present invention require no special driver software in order to communicate because such an application communicates directly with the subsystem by writing to and reading from conventional OS constructs such as partitions and files. This allows the subsystem communication source code to be run on multiple computing platforms (i.e., in a platform-independent manner) simply by recompiling that source code. No rewriting of the subsystem communication source code is required, as must often be done with current techniques. And, unlike current techniques, no new driver software need be created to operate an application's subsystem communication code on a new computing platform.

Detailed Description Text (5):
An application according to the present invention is configured to run on a host computer 10, and permit host computer 10 to communicate with one or more of its subsystems using the method of the present invention. Host computer system 10 includes a bus 12 which interconnects major subsystems such as a central processing unit (CPU) 14, a system memory 16 (e.g., dynamic random-access memory or DRAM), an input/output (I/O) adapter 18, an external device such as a display screen 24 via

display adapter 26, a keyboard 32 and a mouse 34 via I/O adapter 18, and a subsystem controller 36. A floppy disk drive 38 operative to receive a floppy disk 40 is also provided.

Detailed Description Text (17):
A primary advantage of this technique is the ability to send information to and from a subsystem without the need for platform-specific driver software. This is a departure from the conventional approach in which platform-specific driver software is written specifically to allow application software to communicate with the subsystem's controller. Because the present invention uses standard drivers (i.e., those that are provided with a given operating system or with the interface controller), application program 100 is platform independent, and need only be recompiled to run on other platforms (assuming the portions of application program 100 other than those of the present invention are similarly portable). Because the present invention obviates the need for the platform-specific driver software required by conventional methods, no driver installation procedure need be performed, simplifying application installation and system maintenance. Another advantage of this technique is that if write and read commands are issued in pairs, changes in the identifier between the issuance of these pairs can occur without causing any problems.

Detailed Description Text (37):
These vendor-unique commands can be filtered and interpreted as SDU communication commands. Although it is possible to support SCSI pass-through commands in a platform-specific manner, each implementation's interface would then differ from platform to platform. The SDU of such an approach would have to be ported and tested on each platform supported. This approach may be used to provide a PC-based service tool for off-line analysis of sub-systems without involving the real host system, but is antithetical to the idea of providing platform-independent support.

Detailed Description Text (64):
In the case of an installation in a PC operating system, the same basic procedure is followed, albeit with OS-specific adjustments. For example, the software is loaded onto a system drive and installed using a script or installation procedure. As before, a configuration file can be used to store configuration information. However, in this case a file is created (either by the user or the installation program) that the SSA_SERV software will use to write commands to and read data from, to support the given embodiment of the present invention. As before, multiple files could be used to provide redundancy.

CLAIMS:

1. A method for communicating between a host computer and a subsystem controller comprising: sending data to a device driver, wherein said data comprising an identifier and a command, said device driver is configured to communicate said data to said subsystem controller, and said device driver is not configured to recognize said identifier; communicating said identifier and said command from said device driver to said subsystem controller; identifying said identifier by analyzing at least a portion of said data; and identifying said command as a protocol command using said identifier.

23. A computer system comprising: a host computer, comprising a processor; a subsystem controller, coupled to said host computer; computer readable medium coupled to said processor; and computer code, encoded in said computer readable medium, configured to cause said processor to: send data to a device driver, wherein said data comprising an identifier and a command, said device driver is configured to communicate said data to said subsystem controller, and said device driver is not configured to recognize said identifier; communicate said identifier and said command from said device driver to said subsystem controller; identify said identifier by analyzing at least a portion of said data; and identify said

command as a protocol command using said identifier.

24. A computer program product encoded in computer readable media, said computer program product comprising: a first set of instructions, executable on a computer system, configured to send data to a device driver, wherein said data comprising an identifier and a command, said device driver is configured to communicate said data to said subsystem controller, and said device driver is not configured to recognize said identifier; a second set of instructions, executable on said computer system, configured to communicate said identifier and said command from said device driver to said subsystem controller; a third set of instructions, executable on said computer system, configured to identify said identifier by analyzing at least a portion of said data; and a fourth set of instructions, executable on said computer system, configured to identify said command as a protocol command using said identifier.

25. An apparatus for communicating between a host computer and a subsystem controller comprising: means for sending data to a device driver, wherein said data comprising an identifier and a command, said device driver is configured to communicate said data to said subsystem controller, and said device driver is not configured to recognize said identifier; means for communicating said identifier and said command from said device driver to said subsystem controller; means for identifying said identifier by analyzing at least a portion of said data; and means for identifying said command as a protocol command using said identifier.

30. The apparatus of claim 28, wherein said data line is further configured to carry data from said host computer to said device.

First Hit Fwd Refs

Generate Collection Print

L8: Entry 3 of 5 File: USPT Dec 30, 2003

DOCUMENT-IDENTIFIER: US 6671882 B1
TITLE: System for distributing and handling electronic program guide information using CORBA-wrapped objects

Abstract Text (1):
A translator for converting items of interactive program guide data to data structures that are more universal to handle with popular platforms, operating systems, tools, utilities and other hardware and software processors and resources. The invention uses C++ class objects and structures. The objects and structures are placed into a Common Object Request Broker Architecture (CORBA) "wrapper." This allows the objects to be handled by platform-independent interfaces so that the system is easily adaptable to different hardware devices and software functionality. Aspects of the invention include the translation from custom IPG formats to CORBA-wrapped C++ objects, the storage of the objects, transmission of the objects among devices, and data entry and error handling of information represented by the objects.

Brief Summary Text (11):
The present invention converts items of IPG data to data structures that are more universal to handle with popular platforms, operating systems, tools, utilities and other hardware and software processors and resources. The invention uses C++ class objects and structures. The objects and structures are placed into a Common Object Request Broker Architecture (CORBA) "wrapper." This allows the objects to be handled by platform-independent interfaces so that the system is easily adaptable to different hardware devices and software functionality. Aspects of the invention include the translation from custom IPG formats to CORBA-wrapped C++ objects, the storage of the objects, transmission of the objects among devices, and data entry and error handling of information represented by the objects.

Detailed Description Text (11):
IPGT 142 outputs "demand" 164 and "trickle" 166 data streams for eventual broadcast, or other transfer, to a consumer's IRD 102, as shown in FIG. 1. The demand data stream is a high-speed real-time data stream. All demand IPG data block records needed for a given time frame are found in that time frame's IPG data block. Demand blocks are page addressed and the IRD can fetch all the IPG data pages it requires. An IPG Demand Service can be configured to have a maximum output data rate of 1.5 Mbps. The trickle data stream is a lower-speed data stream that drops into the IRD as a background process and is stored in the IRD's memory. Duplicate schedule and description records are removed to reduce the total data size. The trickle data files can be FTP'd to the SAC for delivery or sent to the Encoder's SAC port directly (as shown in FIG. 1). The maximum data rate of the trickle files is 200 Kbps.

Detailed Description Text (18):
IPGT 142 also provides a monitoring, editing and control interface via user interface 150, event listener 152 and configuration server 158. User interface 150 is a graphical user interface (GUI) to the IPGT system. It enables an operator to control the startup of the entire system and verify that the system is functioning properly. Also, the IPGT configuration and data can be modified through the user interface.

Detailed Description Text (19):
Configuration Server 158 provides information to each of the IPGT subsystems to control the generation and transmission of IPG data. The Configuration Server acts as a layer between configuration settings selected through the GUI and the rest of the system.

Detailed Description Text (37):
In FIG. 4B, subsystems within box 220 are directly interfaced to internal bus 228. Such subsystems typically are contained within the computer system such as within cabinet 206 of FIG. 4A. Subsystems include input/output (I/O) controller 222, System Memory (or "RAM") 224, CPU 226, Display Adapter 230, Serial Port 240, Fixed Disk 242, Network Interface Adapter 244. The use of bus 228 allows each of the subsystems to transfer data among subsystems and, most importantly, with the CPU. External devices can communicate with the CPU or other subsystems via bus 228 by interfacing with a subsystem on the bus. Thus, Monitor 246 connects with Display Adapter 230, a relative pointing device (e.g. a mouse) connects through Serial Port 240. Some devices such as Keyboard 250 can communicate with the CPU by direct means without using the main data bus as, for example, via an interrupt controller and associated registers.

First Hit Fwd Refs

☐ Generate Collection Print

L8: Entry 3 of 5 File: USPT Dec 30, 2003

US-PAT-NO: 6671882
DOCUMENT-IDENTIFIER: US 6671882 B1

TITLE: System for distributing and handling electronic program guide information using CORBA-wrapped objects

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Murphy; Pat	San Diego	CA		
Crofts; Cathy	Oceanside	CA		
Mahan; Jason	San Diego	CA		
Hopper; Jeff	San Diego	CA		
Harrington; John	Encinitas	CA		
Datte; Chuck	Encinitas	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
General Instrument Corporation	Horsham	PA			02

APPL-NO: 09/ 359169 [PALM]
DATE FILED: July 21, 1999

PARENT-CASE:
CLAIM OF PRIORITY This application claims priority from U.S. Provisional Patent Application Serial No. 60/094,037 filed on Jul. 25, 1998 which is hereby incorporated by reference as if set forth in full in this document.

INT-CL: [07] H04 N 5/45, G06 F 3/00

US-CL-ISSUED: 725/54; 345/700, 709/203, 712/300
US-CL-CURRENT: 725/54; 345/700, 709/203, 712/300

FIELD-OF-SEARCH: 725/39, 725/48, 725/54, 345/700, 709/200, 712/300

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5923879	July 1999	Sasmazel et al.	717/143

☐	5983233	November 1999	Potonniee	707/103R
☐	6049819	April 2000	Buckle et al.	709/202

OTHER PUBLICATIONS

Baranitharan Subbiah, Rajesh Chandrasekhar, Thomas Kuehnel, and Gottfried W.R. Luderer Video on Demand in a Distributed Environment using Corba 1996 Conf. 31 pp. 321-330.*
Saleem N. Bhatti and Graham Knight, University College London On Management of CATV Full Service Networks: A European Perspective IEEE Sep./Oct. 1998 pp. 28-39.

ART-UNIT: 2611

PRIMARY-EXAMINER: Faile; Andrew

ASSISTANT-EXAMINER: Nalevanko; Chris

ATTY-AGENT-FIRM: Kulas; Charles J. Townsend and Townsend and Crew, LLP

ABSTRACT:

A translator for converting items of interactive program guide data to data structures that are more universal to handle with popular platforms, operating systems, tools, utilities and other hardware and software processors and resources. The invention uses C++ class objects and structures. The objects and structures are placed into a Common Object Request Broker Architecture (CORBA) "wrapper." This allows the objects to be handled by platform-independent interfaces so that the system is easily adaptable to different hardware devices and software functionality. Aspects of the invention include the translation from custom IPG formats to CORBA-wrapped C++ objects, the storage of the objects, transmission of the objects among devices, and data entry and error handling of information represented by the objects.

6 Claims, 6 Drawing figures

L3: Entry 8 of 11 File: USPT Aug 10, 1999

DOCUMENT-IDENTIFIER: US 5937193 A
TITLE: Circuit arrangement for translating platform-independent instructions for execution on a hardware platform and method thereof

Abstract Text (1):
A translating circuit coupled to a processor and memory of a computer system translates platform-independent instructions such as Java bytecodes into corresponding native instructions for execution by the processor. In one embodiment, the translating circuit is incorporated into the same integrated circuit device as the processor. In another embodiment, the translating circuit is provided within one or more external integrated circuit devices. One or more look-up tables map platform-independent instructions into one or more native instructions for the processor, thereby minimizing software-based interpretation of platform-independent program code. Moreover, platform-independent instructions are mapped to native instructions on-the-fly, or alternatively, in blocks prior to execution using a state machine.

Brief Summary Text (4):
Platform-independent programming languages, such as the "Java" programming language from Sun Microsystems, Inc. offer significant advantages over traditional, platform-specific languages. A platform-independent programming language typically utilizes platform-independent program code (machine-readable instructions) suitable for execution on multiple hardware platforms without regard for the particular instruction set for the hardware platforms. A hardware platform typically includes a computer system having one or more processors (e.g., microprocessors or microcontrollers) which execute a particular set of instructions having a specific format, sometimes referred to as a native instruction set. This is in contrast to platform-specific languages, which utilize platform-specific compilers to generate program code that is native to one particular hardware platform. While the same source code may in some instances be compiled by different platform-specific compilers into suitable program code for multiple platforms, the resulting program code is not platform-independent.

Brief Summary Text (5):
In many environments, platform-independent program codes are in an intermediate code format, since further processing is required to execute such codes on a specific hardware platform. For Java, for example, the intermediate codes are referred to as bytecodes. Typically, a compiler is used to generate a series of intermediate codes from a source file. The intermediate codes are then executed by a software interpreter which converts them into native instructions for the computer system on the fly. Consequently, the intermediate codes are executable on any computer system having a suitable platform-independent program code interpreter.

Brief Summary Text (6):
Many platform-independent program codes are typically relatively compact, which makes them readily suited for downloading over a network or modem. Moreover, since the program code is platform-independent, the downloading computer system (or server) can download the same program code irrespective of the particular hardware platform of the executing computer system (or client). Consequently, platform-

independent program codes such as Java are expected to enjoy immense popularity for the distribution of software programs over the Internet. Typically, platform-independent software programs downloaded from the Internet are in the form of applets which execute within a web browser. It should be understood, however, that platform-independent program codes have many other uses, including in stand-alone applications, operating systems, and real-time embedded systems, among others.

Brief Summary Text (7):
One problem with platform-independent program code, however, is that the program code must be interpreted during run time, which significantly reduces execution speed compared to program code native to a particular hardware platform. Some Java interpreters, for example, may require up to 50 processor clock cycles to process each bytecode, compared to typically one clock cycle for most native instructions.

Brief Summary Text (8):
As an alternative to run time interpretation, software-based just-in-time (JIT) compilers have been developed to optimize interpretation of platform-independent program code, typically by emulating the functionality of the platform-independent code using native code. While execution speed is increased over simple runtime interpretation, the platform-independent program code is still slower than native code, and additional memory space is required to store the compiler code.

Brief Summary Text (9):
At the other extreme, dedicated processors (e.g., for Java, the picoJAVA, microJAVA and UltraJAVA processors from Sun Microelectronics) have been proposed to utilize platform-independent instructions as their native instruction set. While these processors have the capability of running platform-independent program code as fast as other native program codes for other hardware platforms, the processors suffer from the same problems as any other processor when executing non-native program code.

Brief Summary Text (10):
It is estimated that in the future as much as 50% or more of the program code run on any particular hardware platform may be platform-independent. However, a large portion of program code will still be platform specific. Consequently, a substantial need exists for a manner of accelerating the execution of platform-independent program code on a hardware platform without adversely impacting the execution speed of native program code thereon.

Brief Summary Text (12):
In accordance with one aspect of the invention, there is provided a computer system for executing program code in a format having platform-independent instructions. The computer system includes a processor for executing program code in a format having native instructions; a memory coupled to the processor for storing program code for execution by the processor; and a translation circuit coupled to the memory, the translation circuit configured and arranged to receive a platform-independent instruction from the memory and output at least one native instruction for the processor corresponding thereto.

Brief Summary Text (13):
In accordance with another aspect of the invention, there is provided a translation circuit for use in a computer system to execute program code in a format having platform-independent instructions on a processor in the computer system that executes program code in a format having native instructions. The translation circuit includes an input circuit configured and arranged to receive a platform-independent instruction; and translating means, coupled to the input circuit, for outputting at least one native instruction for the processor corresponding the platform-independent instruction.

Brief Summary Text (14):

According to a further aspect of the invention, there is provided a method of executing program code in a format having platform-independent instructions on a processor which executes program code in a format having native instructions. The method includes receiving a platform-independent instruction; and translating the platform independent instruction with a hardware-implemented translation circuit into at least one corresponding native instruction using an object table including a plurality of entries, each entry matching a platform-independent instruction with at least one native instruction for the processor.

Detailed Description Text (3):
It is typically over network 8 that computer system 10 is likely to receive platform-independent program code, since often the servers coupled to computer system 10 over network 8 are unaware of the particular hardware platform of the system, and since it is often desirable for the servers to only have to download one version of program code for a given application. However, it should be appreciated that platform-independent program code may be received from computer system 10 by any number of alternate manners, including removable storage devices such as floppy disks, CD-ROM's, magnetic tape, and flash memories, etc. Moreover, platform-independent program code may be initially stored in computer system 10, e.g., in ROM's and other non-volatile devices, on hard drives, etc. It should also be appreciated that the platform-independent program code executed by computer system 10 may include any type of computer program, including stand-alone applications, operating systems, embedded applications, etc., as well as applets which are executed within a web browser.

Detailed Description Text (4):
Computer system 10 is implemented in other embodiments as a network computer (NC) which has little or no mass storage and which principally executes applications downloaded from server 5. With a network computer, a substantial portion of the program code executed by the computer is typically platform-independent. Thus, that this type of application particularly benefits from the performance enhancements offered by the various embodiments of the present invention.

Detailed Description Text (5):
Computer system 10 may also be any of a number of other computer or data processing systems in which platform-independent program code may be utilized, including various network servers, minicomputers, mainframe computers, workstations, desktop computers, laptop computers, mobile computers, embedded controllers, etc.

Detailed Description Text (6):
Certain embodiments of the invention generally operate by converting platform-independent instructions to be executed by a processor into corresponding native instructions for the processor using a hardware-implemented translation circuit. As will be discussed below, the translation occurs immediately prior to execution by the processor, whereby the translation circuit outputs directly to the processor. In the alternative, the translation circuit outputs to a memory for storing the native instructions. The instructions are then pulled from the memory and executed by the processor (either immediately or at a later instance).

Detailed Description Text (7):
In addition, many of the embodiments of the invention are specifically directed to processing Java bytecodes. However, it should be appreciated that other embodiments may process other platform-independent instruction sets now or hereafter developed consistent with the present invention.

Detailed Description Text (8):
FIG. 2 illustrates a specific embodiment of computer system 10 in greater detail. In this embodiment, a look-up table is utilized to directly map platform-independent instructions to corresponding native instructions.

Detailed Description Text (12):
Processor 40 is configured to operate in one of two modes. A first, native mode is the normal operating mode of the processor, whereby 32-bit native instructions are pulled from system memory 25 and executed directly by the processor. In a second, platform-independent (Java) mode, consistent in operation with a Java-type program code, a translation circuit 50 coupled to system data bus 24 is utilized to receive 8-bit Java bytecodes and output corresponding 32-bit native instructions directly to processor 40 for execution thereon.

Detailed Description Text (13):
The native and platform-independent modes are selected by processor 40 via a JMode mode select signal. In one specific embodiment, the mode select signal is output from a decoder 45 (JDecode) coupled to the address lines 42 of processor 40, such that the platform-independent mode is selected whenever an address within a specified range is placed on address lines 42. This in effect partitions the system memory into native and platform-independent partitions. Other manners of switching modes may also be used, e.g., using a special instruction or IO operation to switch from native to platform-independent mode, and an exception or special instruction to switch back to native mode. In addition, mode switching in some embodiments is selectively disabled (e.g., through user input via the keyboard or other user input device) to disable the translation circuit.

Detailed Description Text (15):
Input lines 52, byte select multiplexer 56 and table program multiplexer 58 together form an input circuit for the translation circuit through which platform-independent instructions are received. The input circuit optionally includes a register or other storage device for storing the platform-independent instruction prior to processing. Moreover, either of the multiplexers may be omitted from the input circuit, e.g., if the platform-independent instructions are the same width as the data bus, or if the table is implemented in non-volatile memory and is not programmable by processor 40. Thus, the input circuit in some embodiments includes only a set of input lines (e.g., address, data or control lines), or a port, depending on the application. Other modifications to the input circuit are possible.

Detailed Description Text (16):
Object table 51 forms a translating means for translation circuit 50 and is implemented in one embodiment as a 256.times.33 bit RAM having eight input lines 52 (which in this embodiment are address lines) and thirty three output (data) lines 54a and 54b, resulting in 256 possible entries of 33 bits each. Input lines 52 are normally coupled to the output lines of multiplexer 56, while 32 of the data lines (lines 54a) are output to one input of a 32-bit wide 2-to-1 data bus multiplexer 48. The other input of multiplexer 48 is coupled to the system data bus 24, while the output is coupled to data lines 44 of processor 40. One of the two inputs to multiplexer 48 is coupled to processor 40 in response to the JMode mode select signal. Consequently, in native mode, the processor is directly coupled to the system data bus, and in platform-independent mode, the processor is coupled to the system data bus through translation circuit 50.

Detailed Description Text (17):
Each platform-independent instruction is matched with an entry in object table 51 by using the bytecode for the instruction as the index to the table. Each entry therefore includes a 32-bit native instruction that corresponds to the platform-independent instruction which addresses the entry.

Detailed Description Text (21):
The address lines 42 of processor 40 are coupled to the system address bus 22 through an address bus multiplexer 46 which is controlled via the JMode mode select signal. In the native mode, address lines 42 are coupled directly to the system address bus 22. However, in platform-independent mode, the low two order address

lines AA[1:0] are coupled to byte select multiplexer 56 and operate as the byte select signal, and the remaining address lines AA[31:2] are coupled to system address bus lines RA[29:0], with the two high order system address bus lines RA [31:30] coupled to logic zero. This permits the address of the processor to increment by 4 for every 32-bit operation while only incrementing the address of system memory 25 by 1, thereby permitting four bytecodes to be stored in 32-bit words in system memory 25.

Detailed Description Text (23):
With this configuration, a copy of the contents of object table 51 are maintained in the system memory, e.g., in a dedicated non-volatile area in ROM 28. The contents may also be stored at various times in RAM 26, or on an external storage device, for example. Processor 40 then is able to copy the contents of the table from the ROM, RAM or external device to the object table by alternating read and write operations. Any write operation to the address partition allocated to the object table is decoded by decoder 45 to assert the program mode select signal, thereby coupling input lines 52 of table 51 to system address bus 22 and placing table 51 into a write mode. With the data lines 54a of table 51 coupled to system data bus 24, processor 40 is then able to write appropriate information to the table.

Detailed Description Text (26):
However, when processor 40 requests instructions from a platform-independent partition of the system memory, the mode select signal is asserted by decoder 45, which couples the low two order address lines AA[1:0] of address lines 42 to byte select multiplexer 56, couples address lines AA[0,0,31:2] to system address bus lines RA[31:0] and couples data lines 44 to data lines 54a of object table 51.

Detailed Description Text (28):
After execution of a native instruction when in the platform-independent mode, processor 40 increments its address counter to the next instruction, which has the effect of selecting the next bytecode provided to byte select multiplexer 56 over system data bus 24. It should be noted that, in this arrangement, the address accessed in system memory 25 increments by one for every four increments of the address counter in processor 40 to reflect the presence of four bytecodes in each 32-bit word stored in the platform-independent partition in the system memory.

Detailed Description Text (29):
Return from platform-independent mode to native mode occurs whenever processor 40 accesses the native partition of system memory 25. Consequently, switching between modes may be performed automatically and without dedicated processing by processor 40.

Detailed Description Text (30):
It should be appreciated that the particular design of translation circuit 50 may vary significantly depending upon factors such as the width of the platform-independent instructions, the width of the native instructions, and the mapping required between the two. For example, if the platform-independent and native instructions are the same width, no separate byte select multiplexer is required.

Detailed Description Text (32):
It is believed that a greater number of Java bytecodes could be mapped directly for execution on the aforementioned CISC processors, and thus the direct mapping technique disclosed herein may provide comparatively greater performance enhancements. However, it should also be noted that a platform-independent instruction set could be developed based upon a store-and-forward virtual machine architecture, thereby making the direct mapping technique exceptionally suitable for accelerating the operation of an ARM processor or similar device executing such program code.

Detailed Description Text (33):
Other mapping techniques may be utilized to map a greater percentage of platform-independent instructions to native instructions and thereby accelerate performance and minimize exceptions.

Detailed Description Text (35):
As another example, a two level or double look-up technique is used to further increase the percentage of mappable platform-independent instructions. In one embodiment, a first, or vector, table maps platform-independent instructions to a vector, or an address, in a second, or object, table storing anywhere from 0 to n native instructions for each bytecode. The first table is implemented in a small RAM (e.g., a 200.times.8 bit RAM, since there are approximately 200 valid Java bytecodes), with the second table implemented in a 256.times.34 bit RAM. The address in the first table points to the starting address in the second table for a sequence of native instructions, essentially providing a microcoded routine of native instructions for implementing a bytecode. Again, a separate bit (e.g., bit 34) is used to indicate that at least one more native instruction must be executed after a given table entry is processed. Also, exception conditions are designated by a separate bit (e.g., bit 33). Any of the above alternatives, such as utilizing a dedicated native instruction to signal an exception, may also be used.

Detailed Description Text (36):
Another computer system 100 consistent with the invention is illustrated in FIG. 3, including a microprocessor circuit 140 (e.g., an ARM processor) coupled to a system memory 125 (including RAM 126 and ROM 128) over an address bus 122 and data bus 124. A translation circuit 150, having a master interface 152 and slave interface 154, is also coupled to address bus 122 and data bus 124. Master interface 152 permits translation circuit 150 to operate as a bus master to translate platform-independent program code stored in memory 125 into corresponding native program code and store the native code back into memory 125. Master interface 152 requests control of the bus from processor 140 through a REQ line, and obtains control when processor 140 asserts a GRANT line. Slave interface 154 is used by processor 140 to program initial memory addresses used by the translation circuit for the code translation.

Detailed Description Text (37):
Master interface 152 in this embodiment operates as a block processing means which permits fill blocks of platform-independent program code to be translated and stored back into memory 125, rather than being supplied directly to the processor. It should be appreciated that other block processing mechanisms which essentially automate the passage of multiple platform-independent instructions through the translating means may be used in the alternative.

Detailed Description Text (39):
Translation circuit 150 primarily operates as a two level look-up table, utilizing a first, vector table, and a second, object table. The first table essentially provides a starting address (vector) in the second table for a microcoded routine of one or more native instructions that correspond to a given platform-independent instruction. However, in this embodiment, the tables themselves are stored in system memory 125, whereby the translating means for the translation circuit essentially includes registers which point to vectors or entries in the tables, rather than the actual tables.

Detailed Description Text (40):
As shown in FIG. 4, translation circuit 150 requires several memory pointers which are stored in registers controlled via a translation state machine 153 in master interface 152. A source address register 160 points to the address of the platform-independent program code to be translated, and a destination address register 164 points to the address to which corresponding native instructions are to be stored by the circuit. The platform-independent instruction stored at the current address

pointed to by register 170 is received over data bus 124 by instruction register 170.

Detailed Description Text (42):
A translation vector base address register 172 stores the starting address of the first table, which is essentially a table of vectors pointing to the code translations. A translation vector address calculation block 174 receives the platform-independent instruction and the translation vector base address from blocks 170, 172 and outputs the address of a vector in the second table that points to the native code corresponding to the platform-independent instruction. In one embodiment, block 174 is implemented as a summer which adds together the values in registers 170, 172. Also, the value in register 170 is first shifted left two bits to convert the platform-independent code into a 32-bit offset value. Block 174 also includes in some embodiments a separate register to store the address pointing to the vector. In addition, a translate object address register 176 stores the vector stored at the address output from block 174, and is incremented as native code instructions are stored to the system memory starting at the destination address.

Detailed Description Text (44):
The reserved codes are unused values that do not represent any valid native instructions. One reserved code is an END code that indicates the end of a sequence of native code to in effect delimit entries in the second table. Another reserved code is an IMM code that enables embedded immediate data in the platform-independent code to be passed through the translation circuit (as discussed below). An additional reserved code is an EXC code that indicates an exception condition to halt the translation process and interrupt the processor, thereby allowing the processor to interpret any functions that the translation circuit is not capable of translating. Other reserved codes may also be used.

Detailed Description Text (45):
The dataflow involved in the translation of a platform-independent instruction into a native instruction is illustrated in FIG. 5. Generally, source address register 160 points to a platform-independent instruction in a block of source platform-independent code 190 in memory 125. The platform-independent instruction is stored in instruction register 170, which is added to the contents of translation vector base address register 172, resulting in a pointer to a vector in translation vector table 192. The vector is stored in translate object address register 176, and is used to point to corresponding native code stored in translation object table 194. Native instructions in table 194 are stored in temporary data register 178, and subsequently into memory 125 at the address pointed to by destination address pointer 164 to form a block of output code 196.

Detailed Description Text (48):
FIG. 6 illustrates an exemplary program flow of one specific embodiment, designated Translate Code routine 200, for translation state machine 153. Prior to executing routine 200, processor 140 initializes source address register 160 and source end address register 162 to point to the starting and ending addresses, respectively, of a block of platform-independent program code to be translated. Processor 140 also initializes destination address register 164 to point to the area to which the translated code is to be stored. Routine 200 is started in translation circuit 150 in response to the GRANT line being asserted by processor 140 (see FIG. 3), whereby control over bus 122, 124 is handed over to bus master 152.

Detailed Description Text (49):
First, in block 202 a platform-independent instruction (here a Java bytecode) is fetched from the memory location pointed to by source address register 160 and stored in instruction register 170. Next, in block 204 an index into table 192 is calculated by block 174 by summing the translation vector base address from register 172 with the contents of instruction register 170 shifted left two bits. Next, in block 206, the translate vector stored in table 192 at the address pointed

to by the table index (Translate Vector Address) is stored in translate object address register 176. At this point, register 176 contains the starting address of the native code in table 194 corresponding to the platform-independent instruction being translated.

Detailed Description Text (52):
If, however, the instruction is an END reserved code, block 222 passes control to block 224 to determine if the current source address stored in register 160 is equal to the source end address stored in register 162 by testing if translation complete signal 168 has been asserted by block 166. If the source address does not equal the source end address, additional platform-independent instructions must be processed, and control passes to block 226 to increment source address register 160. Then, control returns to block 202 to process the next platform-independent instruction.

Detailed Description Text (54):
Returning to block 210, translation state machine 153 handles exception conditions by testing for an EXC code. Upon receipt of this code, control is passed to block 212 where the translation process is halted, the REQ signal is released, and an interrupt is sent to processor 140. The exception condition is generally implemented when a platform-independent instruction cannot or will not be translated by translation circuit 150, thereby enabling processor 140 to interpret the instruction. An example of a Java bytecode that may generate an exception is the "ldcl" bytecode (12h), which pushes an item, which may have varying lengths (e.g., a string), onto the stack. The translation circuit shown herein does not include any mechanism to determine the length of an item, and consequently, processor 140 handles this instruction.

Detailed Description Text (56):
With the aforementioned translation circuit, platform-independent instructions may be translated into any number of native instruction streams. The native instruction streams may include a single native instruction, a sequence of native instructions, a single native instruction with embedded data, a sequence of native instructions with embedded data, an exception, an exception with embedded data, etc.

CLAIMS:

1. A computer system for executing program code in a format having platform-independent instructions, the computer system comprising:

a processor for executing program code in a format having native instructions;

a memory coupled to the processor for storing program code for execution by the processor, wherein the memory is a first memory and is coupled to the processor by an address bus and a data bus;

a vector table and an object table, the object table including a plurality of entries, each entry including at least one native instruction corresponding to one of the platform-independent instructions, and the vector table including a plurality of vectors indexed by the platform-independent instructions, each vector pointing to one of the entries in the object table; and

a translation circuit coupled to the memory, the translation circuit configured and arranged to receive a platform-independent instruction from the memory and output at least one native instruction for the processor corresponding thereto, wherein the translation circuit further comprises:

a state machine, coupled to the memory and the vector and object tables, the state machine configured and arranged to receive a block of platform-independent program code stored in the memory and output a block of corresponding native program code

to the memory;

a source address register configured and arranged to store a source address in the memory of a platform-independent instruction to be translated;

a destination address register configured and arranged to store a destination address in the memory at which to store a native instruction corresponding to the platform-independent instruction pointed to by the source address register;

an instruction register configured and arranged to store the platform-independent instruction pointed to by the source address register; and

a translate object address register configured and arranged to store a translate object address pointing to a native instruction stored in the object table that corresponds to the platform-independent instruction pointed to by the source address register;

wherein the state machine is coupled to the source address register, the destination address register, the instruction register, and the translate object address register, and wherein the state machine is configured and arranged to manipulate the destination address and translate object address registers to store each native instruction, in the entry in the object table corresponding to the platform-independent instruction stored in the instruction register, in the memory starting at the destination address.

2. The computer system of claim 1, wherein entries in the object table include end codes delimiting the end of the entries, and wherein the state machine is configured and arranged to detect the end codes to terminate translation of each platform-independent instruction.

3. The computer system of claim 2, wherein selected entries in the object table further include immediate codes indicating embedded data in the block of platform-independent program code, and wherein the state machine is configured and arranged to detect the immediate codes to store embedded data directly in the corresponding block of native program code.

4. The computer system of claim 1, wherein the platform-independent instruction comprises a Java-type bytecode.

5. The computer system of claim 1, wherein the translation circuit outputs an exception signal to the processor when a platform-independent instruction received by the translation circuit cannot be directly translated to a native instruction.

8. A translation circuit for use in a computer system to execute program code in a format having platform-independent instructions on a processor in the computer system that executes program code in a format having native instructions, the computer system further including a memory, the translation circuit comprising:

an input circuit configured and arranged to receive a platform-independent instruction, the input circuit including a source address register configured and arranged to store a source address of a platform-independent instruction to be translated;

translating means, coupled to the input circuit, for outputting at least one native instruction for the processor corresponding the platform-independent instruction;

a destination address register configured and arranged to store a destination address in the memory for storing a native instruction corresponding to the platform-independent instruction pointed to by the source address register; and

a block processing means, coupled to the source address register, the destination address register, and the translating means, and configured and arranged to supply a block of platform-independent program code stored in the memory to the translating means and store in the memory a block of corresponding native program code output from the translating means.

9. The translation circuit of claim 8, wherein the input circuit includes an instruction register configured and arranged to store the platform-independent instruction, and wherein the translating means includes a translate object address register storing a vector to an entry in an object table in the memory, the entry including at least one native instruction corresponding to the platform-independent instruction.

10. The translation circuit of claim 9, wherein the translating means further comprises a translation vector base address register configured and arranged to store a starting address for a vector table in the memory, the vector table including a plurality of vectors, each of which points to an entry in the object table, and the vector table indexed by the platform-independent instruction stored in the instruction register.

11. The translation circuit of claim 8, wherein the translating means comprises an object table including a plurality of entries, each entry matching a platform-independent instruction with at least one native instruction for the processor.

12. The translation circuit of claim 11, wherein the translating means further comprises a vector table, indexed by the platform-independent instruction received by the receiving means, the vector table including a plurality of vectors, each of which points to an entry in the object table.

17. The translation circuit of claim 8, wherein the block processing means includes a state machine, the state machine acting as a bus master that controls the bus during translation of the block of platform-independent program code.

18. The translation circuit of claim 8, wherein the platform-independent instruction is a Java bytecode.

19. A computer system for executing program code in a format having platform-independent instructions, the computer system comprising:

a processor for executing program code in a format having native instructions;

a memory coupled to the processor for storing program code for execution by the processor;

a translation circuit coupled to the memory, the translation circuit configured and arranged to receive a platform-independent instruction from the memory and output at least one native instruction for the processor corresponding thereto;

a source address register configured and arranged to store a source address in the memory of a platform-independent instruction to be translated;

a destination address register configured and arranged to store a destination address in the memory for storing a native instruction corresponding to the platform-independent instruction pointed to by the source address register; and

a state machine, coupled to the memory, the source address register, the destination address register, and the translation circuit, the state machine configured and arranged to supply a block of platform-independent program code stored in the memory to the translation circuit and store in the memory a block of corresponding native program code output from the translation circuit.

20. The computer system of claim 19, further comprising a vector table and an object table, the object table including a plurality of entries, each entry including at least one native instruction corresponding to one of the platform-independent instructions, and the vector table including a plurality of vectors indexed by the platform-independent instructions, each vector pointing to one of the entries in the object table.

21. The computer system of claim 20, wherein entries in the object table include end codes delimiting the end of the entries, and wherein the state machine is configured and arranged to detect the end codes to terminate translation of each platform-independent instruction.

22. The computer system of claim 21, wherein selected entries in the object table further include immediate codes indicating embedded data in the block of platform-independent program code, and wherein the state machine is configured and arranged to detect the immediate codes to store embedded data directly in the corresponding block of native program code.

25. The computer system of claim 19, wherein the platform-independent instruction comprises a Java-type bytecode.

26. The computer system of claim 19, wherein the translation circuit outputs an exception signal to the processor when a platform-independent instruction received by the translation circuit cannot be directly translated to a native instruction.

27. A method of executing program code in a format having platform-independent instructions on a processor which executes program code in a format having native instructions, the method comprising:

receiving a platform-independent instruction, including receiving a block of platform-independent program code;

storing a source address of a platform independent instruction from the block of platform-independent program code in a source address register;

translating the block of platform independent program code with a hardware-implemented translation circuit into a block of corresponding native program code including at least one corresponding native instruction using an object table including a plurality of entries, each entry matching a platform-independent instruction with at least one native instruction for the processor; and

storing the block of corresponding native program code in the memory starting at a destination address stored in a destination address register.

28. The method of claim 27, wherein translating the platform-independent instruction includes using the platform-independent instruction as an index to a vector table coupled to the object table, the vector table including a plurality of vectors, each of which points to an entry in the object table.

29. The method of claim 27, wherein the platform-independent instruction is a Java bytecode.

First Hit Fwd Refs

Generate Collection Print

L3: Entry 8 of 11 File: USPT Aug 10, 1999

US-PAT-NO: 5937193
DOCUMENT-IDENTIFIER: US 5937193 A

TITLE: Circuit arrangement for translating platform-independent instructions for execution on a hardware platform and method thereof

DATE-ISSUED: August 10, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Evoy; David Ross	Tempe	AZ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
VLSI Technology, Inc.	San Jose	CA			02

APPL-NO: 08/ 757430 [PALM]
DATE FILED: November 27, 1996

INT-CL: [06] G06 F 9/45

US-CL-ISSUED: 395/705
US-CL-CURRENT: 717/140; 717/118

FIELD-OF-SEARCH: 364/DIG.1MSFile, 364/DIG.2MSFile, 395/704, 395/705

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4873628	October 1989	Omori	395/705
5692047	November 1997	McManis	395/704
5740441	April 1998	Yellin et al.	395/704

OTHER PUBLICATIONS

Andrew S. Tanenbaum, "Structured Computer Organization" (1984) pp. 10-12, 1984.
The ARM6 Family Bus Interface, Advanced RISC Machines Ltd. (1996).
The ARM Microprocessor Architecture, Advanced RISC Machines Ltd. (1996).
Wayner, "Sun Gambles On Java Chips", Byte, Nov. 1996.

Java Virtual Machine Architecture, Sun Microsystems (1996).

ART-UNIT: 278

PRIMARY-EXAMINER: Harrell; Robert B.

ATTY-AGENT-FIRM: Wood, Herron & Evans, L.L.P.

ABSTRACT:

A translating circuit coupled to a processor and memory of a computer system translates platform-independent instructions such as Java bytecodes into corresponding native instructions for execution by the processor. In one embodiment, the translating circuit is incorporated into the same integrated circuit device as the processor. In another embodiment, the translating circuit is provided within one or more external integrated circuit devices. One or more look-up tables map platform-independent instructions into one or more native instructions for the processor, thereby minimizing software-based interpretation of platform-independent program code. Moreover, platform-independent instructions are mapped to native instructions on-the-fly, or alternatively, in blocks prior to execution using a state machine.

29 Claims, 6 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L1: Entry 1 of 1 File: USPT Oct 2, 2001

DOCUMENT-IDENTIFIER: US 6298354 B1
TITLE: Mechanism and process to transform a grammar-derived intermediate form to an object-oriented configuration database

Abstract Text (1):
A method and system for transforming an intermediate form into an object-oriented database. The intermediate form is derived from a grammatical form of an object-oriented database through the process of compilation. The grammatical form is an expression of an object-oriented database in a textual form according to a grammar. The intermediate form comprises an array of intelligent entry objects that encapsulate data with methods for manipulating that data. The intermediate form comprises entries as in the object-oriented database but lacks the infrastructure of the database. The intermediate form can be used to populate the object-oriented database with entries. Population takes place through a public API for accessing the object-oriented database; in other words, through an interface which declares methods for navigating the database and adding entries to the database. The object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. The object-oriented database is platform-independent and is therefore configured to be hosted on several different operating systems and computing platforms.

Parent Case Text (2):
This application is related to U.S. patent application Ser. No. 09/253,840, filed on Feb. 19, 1999, entitled "AN INTELLIGENT OBJECT-ORIENTED CONFIGURATION DATABASE SERIALIZER"; U.S. patent application Ser. No. 09/253,841, filed on Feb. 19, 1999, entitled "PROCESS FOR TRANSFORMING CONFIGURATION DATABASE GRAMMAR INTO INTERMEDIATE FORM THAT SIMPLIFIES DATABASE GENERATION"; U.S. patent application Ser. No. 09/253,866, filed on Feb. 19, 1999, entitled "GRAMMAR TO REPRESENT A HIERARCHICAL OBJECT-ORIENTED DATABASF"; U.S. patent application Ser. No. 09/253,867, filed on Feb. 19, 1999, entitled "TRANSFORMATION CUSTOMIZER FOR CONFIGURATION DATABASE COMPILATION AND SERIALIZATION PROCESSES"; U.S. patent application Ser. No. 09/253,868, filed on Feb. 19, 1999, entitled "AN INTELLIGENT INTERMEDIATE STATE OF AN OBJECT-ORIENTED DATABASE"; U.S. patent application Ser. Nos. 09/079,501; 09/079,042; 09/079,500; 09/079,103; 09/079,102; 09/079,499; 09/079,043; and 09/107,048, which are all incorporated herein by reference.

Brief Summary Text (5):
Database systems are serving increasingly important roles in today's society. Modern database systems enable users to gather, manipulate, and maintain massive amounts of information. A mere handful of examples of the myriad uses of database systems includes computerized library systems, automated teller machines, flight reservation systems, computerized parts inventory systems, and configuration databases for computer systems and networks.

Brief Summary Text (8):
One specific type of database is a database employed by an operating system to

maintain configuration information that relates to components of software and/or hardware of a computer system. For example, such a configuration database may store configuration information relating to application programs, hardware devices which are coupled to the computer system, and/or elements of the operating system. These configuration databases may be implemented in many different ways. To exploit the advantages of the object-oriented paradigm, configuration databases may be implemented as object-oriented databases. Unfortunately, these configuration databases, object-oriented or otherwise, are associated with the same difficulties as other types of database systems. For instance, if information in a configuration database is generated dynamically upon the start-up of a computer system, then that information will be lost from session to session unless it is stored in a convenient way.

Brief Summary Text (9):
Therefore, it is desirable to provide an intelligent mechanism and process for storing an object-oriented configuration database.

Brief Summary Text (15):
In one embodiment, the object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. In one embodiment, the object-oriented database is a platform-independent one, such as the Java.TM. System Database, and is therefore configured to be hosted on several different operating systems and computing platforms. In one embodiment, database transformation according to the present invention is implemented as a package of classes and interfaces in the object-oriented Java.TM. Language.

Detailed Description Text (7):
As will be described in further detail below, the computer system 100 implements a system and method for transforming a grammar-derived intermediate form to an object-oriented configuration database through the process of database population. In various embodiments, the computer system 100 further implements a database transformation system and method wherein an active object-oriented database is serialized into a persistent form which is described by a database description grammar, wherein the persistent, grammatical form is compiled into an intelligent intermediate form, and wherein serialization and compilation may be modified to accept complex data types. The persistent form may comprise one or more containers. Containers may reside in various forms in the memory 104 on one or more computer systems 100. The database transformation processes may also be referred to as the pushing and pulling of content to and from containers. The pushing and pulling may take place to and from the memory 104, over the network 108, and/or over the I/O bus 110.

Detailed Description Text (8):
As used herein, an object-oriented database is a database, database system, database management system, electronic filing system, or other computerized collection of information which stores items of data as objects. An object typically includes a collection of data along with methods for manipulating that data. In one embodiment, the database is a configuration database. As used herein, a configuration database is a database, database system, database management system, electronic filing system, or other computerized collection of information which stores information relating to the components and/or parameters which characterize a computer system or systems.

Detailed Description Text (26):
The system, method, and storage medium of the present invention are applicable to any object-oriented database. In one embodiment, the object-oriented configuration database is a Java.TM. System Database (JSD), also known as a JavaOS System Database. The JSD is platform-independent but was developed in conjunction with

JavaOS. In other words, the JSD could be hosted on many different operating systems, including JavaOS. The JSD generally allows an operating system, system services, applications, utilities, and other software components to store and retrieve configuration information concerning the software and hardware of a platform, typically a Java.TM.-based platform such as a network computer (NC). Configuration information is arranged to describe, for example, the physical devices that are present in a machine associated with the JSD, the system software services that are installed, and specific user and group application profiles. The JSD serves as a central repository to store, as well as access, substantially any information which is used for configuration purposes.

Detailed Description Text (28):
A large amount of information in the JSD is transient: it does not survive across runtime sessions. The JSD is populated, that is, filled with entries relating to configuration data, during platform initialization. Additional configuration data is added and/or removed when the operating system boots. Transient information must be repopulated from its source into the JSD every time the computer system or operating system boots. Every time JavaOS boots, for example, the JSD is repopulated with information concerning which devices are installed on the platform. The JSD provides a population interface in the Java.TM. Language for adding configuration data from a variety of sources: for example, files, a network, the host operating system, applications, and drivers. In one embodiment, the Java.TM. Language interface for JSD population is named TreePopulator.

Detailed Description Text (29):
The JSD uses a split design: one part resides on a server computer system, and the other part resides on a client computer system. On the server, the configuration information is stored for each user and client machine on the network. At the time the client computer system boots, each client database is populated from the server database with configuration information about a particular machine, group of machines, and machine platform. At the time a user logs in, the client database is populated with configuration information about the user who is logging in and the group of users he or she belongs to, if any.

Detailed Description Text (31):
The Temp namespace is available as temporary storage for both application and system software settings. The Device namespace contains the set of devices available to the local platform. The Interface namespace contains entries that reference services that implement public Java interfaces. The Alias namespace contains entries that reference entries in the Interface namespace and provide friendly naming schemes for existing entries. The Software namespace contains entries for each installed software component. The Config namespace maintains client configuration information and is usually stored on servers.

Detailed Description Text (33):
FIG. 4 is an illustration of a tree structure representing an exemplary Java System Database on a client computer system. The client tree 301 resides on a networked client machine 300 and relates to configuration data of the client computer system 300. The client machine 300 is an example of a computer system 100 as discussed with reference to FIG. 1. The hierarchy of the client tree 301 is manifested using an n-way tree. At the root of the tree is a root entry 302 which does not contain any data. A first level of nodes 304 in client tree 301 collectively define the six standard namespaces as discussed above.

Detailed Description Text (34):
For example, the Software namespace begins at node 306 and includes all nodes and data branching from node 306. All entries in the Software namespace relate to configuration data regarding software applications for the client computer system 300. Entries in the data schema are made up of a unique name, a list of children (entries below the given entry), and a set of tuples. Each tuple contains a

property name and associated property value (i.e., a name-value pair). In a word processing program, for example, a property name can be "font" and the property value can be "Times Roman." Similarly, all entries under the Device namespace 308 are entries that are related to configuration information of the client computer system 300. Every entry in the hierarchy may act as both an entry in a sub-tree and the root of a sub-tree having descendant entries or child nodes. Each namespace in layer 304 is described in U.S. Provisional Application filed on May 14, 1998 and commonly assigned, entitled "JAVA SYSTEM DATABASE", which is incorporated herein by reference.

Detailed Description Text (35):
The Software namespace 306 contains a list of installed and/or available system services such as device drivers, user applications, and user configuration information. The Software namespace 306 includes four categories: application, system, service, and public. In the application category 312, for example, an entry com.Netscape 314 contains the company-unique name "Netscape." Below com.Netscape 314 is an entry 316 for Netscape Navigator, one of Netscape's products. Below the Navigator entry 316 is an entry 318 storing company-specific configuration information relating to Netscape Navigator. The Netscape Navigator application program could access the configuration information 318 while the application program is executing. In a similar way, other application programs could access their own specific configuration entries while executing. Entries 320, 322, and 324 represent other vendors which will also have application-level entries similar to entry 316.

Detailed Description Text (36):
An object-oriented database such as the JSD is largely transient and thus must typically be recreated with every runtime session. Furthermore, the JSD may be difficult to access for data entry and retrieval when it is active in the cache as described above. Therefore, one embodiment of the present invention provides for database transformation from the active, run-time form to a more manageable, persistent form, and then back again. FIG. 5 illustrates an overview of database transformation in accordance with one embodiment of the present invention. The object-oriented configuration database 340 can undergo a process of serialization 342 which transforms the database into a persistent form in one or more containers. The persistent form, also known as a grammatical form 346, is described by a grammar 344. The grammatical form 346 can undergo a process of compilation 348 which transforms the persistent form into an intelligent intermediate form 350. The intelligent intermediate form 350 can be turned back into the object-oriented database 340 through the process of database population 352. A transformation customizer or plug-in 354 can be used to extend the grammar 344 to allow for the use of complex data types in serialization 342 and compilation 348. Although the grammatical form 346 is much smaller than the in-memory object-oriented database 340 due to improved serialization 342, no important information is permanently lost when transforming one into the other, and therefore one form can be transformed into the other form and back again an indefinite number of times.

Detailed Description Text (37):
The JSD is an in-memory repository or cache which can be stored in persistent containers such as files in accordance with the present invention. The JSD defines a public API for dynamically adding, removing, and modifying entries contained within the database. In one embodiment, the public API is a Java.TM. Language interface named TreePopulator. For instance, when the grammatical form is parsed by the configuration tree compiler into an intermediate form, the resulting hierarchical entries of the intermediate form are imported into the JSD using the public API. In this way, the content of the JSD is pushed and pulled from the containers. A single JSD can push and pull content from multiple containers as required to satisfy clients. The JSD pushes and pulls content without regard to container count and implementation. The container count and implementation vary with the complexity and needs of the specific platform. For example, a simple

platform such as a cellular phone may have a single persistent container stored in non-volatile RAM (NVRAM), while a more complex platform such as a network computer may use multiple file-based containers or even an enterprise network directory service such as LDAP (Lightweight Directory Access Protocol).

Detailed Description Text (42):
In one embodiment, the text-based, grammatical description of the contents of the object-oriented database is a static configuration tree. A static configuration tree provides a mechanism whereby hierarchies such as those of an object-oriented configuration database are defined and expressed in a persistent medium using a grammar. The configuration tree is static because each time the grammar description is compiled, it results in the same hierarchy being created within the JSD. Static configuration information is ideal for devices, applications, and services that are simple in nature and do not require dynamic discovery of devices or resources at boot time or service load time.

Detailed Description Text (43):
As used herein, a grammar is a set of rules that govern the structure and/or relationships of terms and/or symbols in a language. As used herein, a grammatical form is a collection of information which is expressed under the rules of a particular grammar. The grammar provided by the present invention is independent of any platform or programming language: the grammar could be used to describe any hierarchical, object-oriented database. A small number of keywords and a simple syntax define the static configuration grammar. Generally speaking, a syntax defines a structure in which the keywords and other elements can be expressed. In other words, a syntax generally relates to the form of the grammar. In one embodiment, the keywords are as follows:

Detailed Description Text (48):
BUSINESS_CARD: Defines a collection of configuration information for an application, device driver, or other software component.

Detailed Description Text (49):
The TREE and ENTRY keywords are used to define the tree root and the tree sub-hierarchy, respectively. The ATTRIBUTES and PROPERTIES keywords, along with name-value pairs, define attributes and properties to be associated with an entry. The scope of a keyword is delimited using opening and closing braces (i.e., curly brackets) "{" and "}". The TREE keyword must appear first because it defines the root of the configuration tree.

Detailed Description Text (50):
FIG. 7 illustrates an example of this hierarchy. A configuration tree 620 is defined using the TREE and ENTRY keywords and the scoping braces of the grammar described above:

Detailed Description Text (63):
The static configuration tree 620 is different in form but identical in content to a configuration database 600. The root 602 has two children, child1604 and child2606. There are three grandchildren 608, 610, and 612, all belonging to child1604. The tree 620 can be transformed into the database 600 through the processes of compilation and/or database population as described in detail below, and the database 600 can be transformed into the tree 620 through the process of serialization as described in detail above. Although the grammar allows the database to be represented in a much more compact form than the in-memory form of the database, no important information is permanently lost in transforming the database to or from the grammatical form. Therefore, the transformations can take place an indefinite number of times back and forth.

Detailed Description Text (126):
The constructor stores these configuration parameters and then allocates a large

scratch array of StaticEntries in preparation to compile the grammatical form to an intermediate form. The StaticEntry objects, which are instances of a StaticEntry class, represent entries in the intermediate form of the database. A StaticEntry object contains references to its parent, siblings, and children, as well as its properties (StaticProperty instances) and attributes (StaticAttribute instances). Furthermore, each StaticEntry contains the entry ID and name. The StaticAttribute, StaticProperty, and StaticEntry classes are defined as follows:

Detailed Description Text (188):
The StaticTreeCompiler produces an intermediate form, not a final form, of the database. The intermediate form lacks the infrastructure of the final database; for instance, the intermediate form is not designed to be accessible by particular application programs for storage of their configuration information. The intermediate form is accessed by the StaticTreePopulator which populates the database. Each StaticTreePopulator has an instance of a compiler and each instance of a compiler has a reference to a StaticTree instance. When the populator is constructed, its constructor is passed an instance of a StaticTree to compile. In one embodiment, therefore, compilation and population are both initiated when a StaticTree instance is passed to a StaticTreePopulator.

CLAIMS:

1. A method for transforming contents of an intermediate form of an object-oriented database into contents of said object-oriented database, said intermediate form and said object-oriented database being stored in a memory of a computer system, said method comprising:

expressing a plurality of entries corresponding to objects in said object-oriented database in said intermediate form, wherein said entries and said objects relate to configuration parameters of said computer system, wherein said intermediate form is derived from a textual form expressed according to a grammar;

populating said object-oriented database with said plurality of entries;

storing said plurality of entries in said object-oriented database in said memory of said computer system, wherein said plurality of entries in said object-oriented database pertain to one or more application programs installed on said computer system.

3. The method of claim 1,

wherein said object-oriented database is configured to be platform independent.

5. The method of claim 1,

wherein said intermediate form includes an array of entries configurable to populate said object-oriented database.

8. A carrier medium comprising program instructions for transforming contents of an intermediate form of an object-oriented database into contents of said object-oriented database, wherein said program instructions are executable to implement:

expressing a plurality of entries corresponding to objects in said object-oriented database in said intermediate form, wherein said entries and said objects relate to configuration parameters of a computer system, wherein said intermediate form is derived from a textual form expressed according to a grammar;

populating said object-oriented database with said plurality of entries; and

storing said plurality of entries in said object-oriented database in a memory of

said computer system, wherein said plurality of entries in said object-oriented database pertain to one or more application programs installed on said computer system.

10. The carrier medium of claim 8,

wherein said object-oriented database is configured to be platform independent.

12. The carrier medium of claim 8,

wherein said intermediate form includes an array of entries configurable to populate said object-oriented database.

15. A computer system for transforming contents of an intermediate form of an object-oriented database into contents of said object-oriented database, said computer system comprising:

a CPU;

a memory coupled to said CPU;

wherein said memory stores said object-oriented database;

wherein said memory stores program instructions executable by said CPU, wherein the program instructions are executable to:

express a plurality of entries corresponding to objects in said object-oriented database in said intermediate form, wherein said entries and said objects relate to configuration parameters of a computer, wherein said intermediate form is derived from a textual form expressed according to a grammar;

populate said object-oriented database with said plurality of entries;

store said plurality of entries in said object-oriented database in said memory, wherein said plurality of entries in said object-oriented database pertain to one or more application programs installed on said computer.

17. The computer system of claim 15,

wherein said object-oriented database is configured to be platform independent.

19. The computer system of claim 15,

wherein said intermediate form includes an array of entries configurable to populate said object-oriented database.

First Hit   Fwd Refs

End of Result Set

Generate Collection   Print

L1: Entry 1 of 1   File: USPT   Oct 2, 2001

DOCUMENT-IDENTIFIER: US 6298354 B1

TITLE: Mechanism and process to transform a grammar-derived intermediate form to an object-oriented configuration database

Abstract Text (1):

A method and system for transforming an intermediate form into an object-oriented database. The intermediate form is derived from a grammatical form of an object-oriented database through the process of compilation. The grammatical form is an expression of an object-oriented database in a textual form according to a grammar. The intermediate form comprises an array of intelligent entry objects that encapsulate data with methods for manipulating that data. The intermediate form comprises entries as in the object-oriented database but lacks the infrastructure of the database. The intermediate form can be used to populate the object-oriented database with entries. Population takes place through a public API for accessing the object-oriented database; in other words, through an interface which declares methods for navigating the database and adding entries to the database. The object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. The object-oriented database is platform-independent and is therefore configured to be hosted on several different operating systems and computing platforms.

Brief Summary Text (11):

The problems outlined above are in large part solved by various embodiments of a method and system for transforming an intermediate form of an object-oriented database into an active object-oriented database in accordance with the present invention. In one embodiment, the intermediate form is derived from a grammatical form of an object-oriented database. A grammatical form, an expression of an object-oriented database in a textual form according to a grammar, may be stored in a persistent form such as one or more files on disk. The grammatical form is human-readable and human-editable. The grammatical form can be created by hand, or it can be created from an object-oriented database in transient form through the process of serialization. The grammar is designed to be platform-independent and programming-language-independent and therefore descriptive of any hierarchical object-oriented database.

Brief Summary Text (15):

In one embodiment, the object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. In one embodiment, the object-oriented database is a platform-independent one, such as the Java.TM. System Database, and is therefore configured to be hosted on several different operating systems and computing platforms. In one embodiment, database transformation according to the present invention is implemented as a package of classes and interfaces in the object-oriented Java.TM. Language.

Detailed Description Text (23):
The porting interface 220 lies below the Java.TM. Virtual Machine 222 and on top of the different operating systems 212b, 212c, and 218 and browser 214. The porting interface 220 is platform-independent. However, the associated adapters 216a, 216b, and 216c are platform-dependent. The porting interface 220 and adapters 216a, 216b, and 216c enable the Java.TM. Virtual Machine 222 to be easily ported to new computing platforms without being completely rewritten. The Java.TM. Virtual Machine 222, the porting interface 220, the adapters 216a, 216b, and 216c, the JavaOS 218, and other similar pieces of software on top of the operating systems 212a, 212b, and 212c may, individually or in combination, act as means for translating the machine language of Java.TM. applications 236, APIs 226 and 230, and Classes 224 and 228 into a different machine language which is directly executable on the underlying hardware.

Detailed Description Text (26):
The system, method, and storage medium of the present invention are applicable to any object-oriented database. In one embodiment, the object-oriented configuration database is a Java.TM. System Database (JSD), also known as a JavaOS System Database. The JSD is platform-independent but was developed in conjunction with JavaOS. In other words, the JSD could be hosted on many different operating systems, including JavaOS. The JSD generally allows an operating system, system services, applications, utilities, and other software components to store and retrieve configuration information concerning the software and hardware of a platform, typically a Java.TM.-based platform such as a network computer (NC). Configuration information is arranged to describe, for example, the physical devices that are present in a machine associated with the JSD, the system software services that are installed, and specific user and group application profiles. The JSD serves as a central repository to store, as well as access, substantially any information which is used for configuration purposes.

Detailed Description Text (124):
The database description grammar is language- and platform-independent. Nevertheless, a database description which follows the rules of the grammar can be compiled into an intermediate form and then used to populate the JSD. As used herein, compilation is any process of transforming information expressed in a first language to information expressed in a second language by applying one or more grammars to interpret and/or analyze the first form and create the second form. Usually, although not always for the purpose of this disclosure, compilation is a process of transforming information expressed in a higher-level language into information expressed in a lower-level language. The lower the level of a language, generally speaking, the easier it is for a computer to understand or execute: in other words, the more readable it is by a computer. The higher the level of a language, generally speaking, the more readable it is by a human.

CLAIMS:

3. The method of claim 1,

wherein said object-oriented database is configured to be platform independent.

10. The carrier medium of claim 8,

wherein said object-oriented database is configured to be platform independent.

17. The computer system of claim 15,

wherein said object-oriented database is configured to be platform independent.

First Hit Fwd Refs

Generate Collection Print

L1: Entry 2 of 8 File: USPT Dec 30, 2003

DOCUMENT-IDENTIFIER: US 6671724 B1
TITLE: Software, systems and methods for managing a distributed network

Detailed Description Text (84):
Referring now to FIGS. 13-16, both the agents and control points may be configured using configuration utility 106. Typically, configuration utility 106 is a platform-independent application that provides a graphical user interface for centrally managing configuration information for the control points and agents. To configure the control points and agents, the configuration utility interface with administrator module 134 of agent 70 and with administrator module 168 of control point 72. Alternatively, configuration utility 106 may interface with administrator module 168 of control point 72, and the control point in turn may interface with administrator module 134 of agent 70.

First Hit Fwd Refs

Generate Collection Print

L1: Entry 2 of 8 File: USPT Dec 30, 2003

US-PAT-NO: 6671724
DOCUMENT-IDENTIFIER: US 6671724 B1

TITLE: Software, systems and methods for managing a distributed network

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pandya; Suketu J.	Mission Viejo	CA		
Lakshmanan; Hariharan	Sunnyvale	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Centrisoft Corporation	Portland	OR			02

APPL-NO: 09/ 532101 [PALM]
DATE FILED: March 21, 2000

INT-CL: [07] G06 F 15/173

US-CL-ISSUED: 709/226; 709/224, 370/236
US-CL-CURRENT: 709/226; 370/236, 709/224

FIELD-OF-SEARCH: 709/223, 709/224, 709/225, 709/226, 709/232, 709/229, 709/234, 709/250, 370/395, 370/236, 710/17, 710/40, 714/20

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5461611	October 1995	Drake, Jr. et al.	
5517622	May 1996	Ivanoff et al.	
5548579	August 1996	Lebrun et al.	
5634006	May 1997	Baugher et al.	
5644715	July 1997	Baugher	
5655081	August 1997	Bonnell et al.	
5657452	August 1997	Kralowetz et al.	

☐	5689708	November 1997	Regnier et al.	709/229
☐	5694548	December 1997	Baugher et al.	
☐	5696486	December 1997	Poliquin et al.	
☐	5706437	January 1998	Kirchner et al.	
☐	5734865	March 1998	Yu	709/250
☐	5761428	June 1998	Sidey	
☐	5774656	June 1998	Hattori et al.	
☐	5777549	July 1998	Arrowsmith et al.	
☐	5781703	July 1998	Desai et al.	
☐	5793958	August 1998	Clement et al.	
☐	5794073	August 1998	Ramakrishnan et al.	710/40
☐	5799002	August 1998	Krishnan	370/234
☐	5822521	October 1998	Gartner et al.	
☐	5848266	December 1998	Scheurich	713/503
☐	5881313	March 1999	Ramakrishnan et al.	710/40
☐	5889958	March 1999	Willens	
☐	5892754	April 1999	Kompella et al.	
☐	5901142	May 1999	Averbuch et al.	
☐	5903568	May 1999	Tanaka et al.	
☐	5907324	May 1999	Larson et al.	
☐	5941947	August 1999	Brown et al.	
☐	5944783	August 1999	Nieten	
☐	5944795	August 1999	Civanlar	
☐	5948065	September 1999	Eilert et al.	709/226
☐	5953338	September 1999	Ma et al.	370/395
☐	5956482	September 1999	Agraharam et al.	
☐	5958010	September 1999	Agarwat et al.	709/224
☐	5968116	October 1999	Day, II et al.	
☐	5978594	November 1999	Bonnell et al.	710/17
☐	5983261	November 1999	Riddle	709/204
☐	5987611	November 1999	Freund	
☐	5996010	November 1999	Leong et al.	
☐	6018567	January 2000	Dulman	
☐	6085241	July 2000	Otis	709/223
☐	6097722	August 2000	Graham et al.	370/395
☐	6125390	September 2000	Touboul	709/223
☐	6199124	March 2001	Ramakrishnan et al.	710/40
	6279039	August 2001	Bhat et al.	709/226

6400687	June 2002	Davison et al.	370/236
6412000	June 2002	Riddle et al.	709/224
6466978	October 2002	Mukherjee et al.	709/225
6466980	October 2002	Lumnelsky et al.	709/226
6502131	December 2002	Vaid et al.	709/224

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 95/27249	October 1995	WO	
WO 97/48214	December 1997	WO	
WO 98/26541	June 1998	WO	
WO 99/34544	July 1999	WO	

OTHER PUBLICATIONS

Moharpatra et al, ATM switching systems for broadband ISDN, Apr. 1997.*
Tsiotsios et al, ATM access-shaping and bandwidth-tuning of DQDB egress traffic, Sep. 1996.*
Orphanos et al., "An Integrated Application/Service Platform to Support Multimedia Applications," IEEE, pp. 1722-1726 (1994).
Lombardo et al., "A Model for Multimedia Service Creation and Activation," IEEE, pp. 1727-1733 (1994).
Nahrstedt et al., "The QoS Broker," IEEE, pp. 53-67 (1995).
Buddenberg, "Session Layer Requirements for Multicastinternets," IEEE, pp. 969-972 (1997).
Nishio et al., "Simplified Method for Session Coordination Using Multi-Level QoS Specification and Translation," IFIP, Ch. 38, pp. 333-344 (1997).
Youssef et al., "The Software Architecture of a Distributed Quality of Session Control Layer," IEEE, pp. 21-28 (1998).
Youssef et al., "Controlling Quality of Session in Adaptive Multimedia Multicast Systems," IEEE, pp. 160-167 (1998).
Youssef et al., "Configurable Multi-Agent System for QoS Control in WATM," IEEE, pp. 2882-2887 (1998).
Hong et al., "A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications," IEEE Network, pp. 70-79 (Mar./Apr. 1999).
Candan et al., "Least-Cost High-Quality Object Retrieval for Distributed Multimedia Collaborations," IEEE, pp. 649-654 (1999).
Cao et al., "An Architecture of Distributed Media Servers for Supporting Guaranteed QoS and Media Indexing," IEEE, pp. 1-5 (1999).
Cheong et al., "QoS Specification and Mapping for Distributed Multimedia Systems: A Survey of Issues," The Journal of Systems and Software, pp. 127-139 (1999).
Koga et al., "Interworking Architecture for Seamless Service Utilization between the TINA-Based Public Network and the Internet," 3529 SPIE 154-65 (1998).
Guedes et al., "An Agent-Based Approach for Supporting Quality of Service in Distributed Multimedia Systems," 21 Computer Communications 1269-78 (1998).
Ott et al., "An Architecture for Adaptive QoS and its Application to Multimedia Systems Design," 21 Computer Communications 334-49 (1998).
Li et al., "End-to-End QoS Support for Adaptive Applications Over the Internet," 3529 SPIE 166-76 (1998).
Alfano, "Design and Implementation of a Cooperative Multimedia Environment with QoS Control," 21 Computer Communications 350-61 (1998).

Almesberger et al., "Quality of Service Renegotiations," 3529 SPIE 124-31 (1998).

ART-UNIT: 2142

PRIMARY-EXAMINER: Powell; Mark R.

ASSISTANT-EXAMINER: Vu; Thong

ATTY-AGENT-FIRM: Kolisch Hartwell, P.C.

ABSTRACT:

Software, systems and methods for managing a distributed network environment including a plurality of computers interconnected by a network link, where at least some of the computers include a layered communications protocol stack for providing a data interface between an application program and the network link, the communications stack having a transport protocol layer for providing an end-to-end communications connection. The invention includes a control module and a plurality of agent modules, each agent being associated with one of the computers and adapted to dynamically monitor the associated computer at a data transmission point between an application program running on the computer and the transport protocol layer and repeatedly communicate with the control module in order to effect management of the distributed network system. The invented software, systems and methods may also include a messaging feature for providing users, IT personnel, or various management systems with informative messages concerning network conditions and network resources.

7 Claims, 19 Drawing figures

L1: Entry 3 of 8 File: USPT Aug 26, 2003

DOCUMENT-IDENTIFIER: US 6611844 B1
TITLE: Method and system for java program storing database object entries in an intermediate form between textual form and an object-oriented form

Detailed Description Text (25):
The system, method, and storage medium of the present invention are applicable to any object-oriented database. In one embodiment, the object-oriented configuration database is a Java.TM. System Database (JSD), also known as a JavaOS System Database. The JSD is platform-independent but was developed in conjunction with JavaOS. In other words, the JSD could be hosted on many different operating systems, including JavaOS. The JSD generally allows an operating system, system services, applications, utilities, and other software components to store and retrieve configuration information concerning the software and hardware of a platform, typically a Java.TM.-based platform such as a network computer (NC). Configuration information is arranged to describe, for example, the physical devices that are present in a machine associated with the JSD, the system software services that are installed, and specific user and group application profiles. The JSD serves as a central repository to store, as well as access, substantially any information which is used for configuration purposes.

First Hit Fwd Refs

Generate Collection Print

L1: Entry 3 of 8 File: USPT Aug 26, 2003

US-PAT-NO: 6611844
DOCUMENT-IDENTIFIER: US 6611844 B1

TITLE: Method and system for java program storing database object entries in an intermediate form between textual form and an object-oriented form

DATE-ISSUED: August 26, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Saulpaugh; Thomas E.	San Jose	CA		
Slaughter; Gregory L.	Palo Alto	CA		
Traversat; Bernard A.	San Francisco	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Sun Microsystems, Inc.	Santa Clara	CA			02

APPL-NO: 09/ 253868 [PALM]
DATE FILED: February 19, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is related to U.S. patent application No. 09/253,839, filed on Feb. 19, 1999, entitled "MECHANISM AND PROCESS TO TRANSFORM A GRAMMAR-DERIVED INTERMEDIATE FORM TO AN OBJECT-ORIENTED CONFIGURATION DATABASE"; U.S. patent application No. 09/253,840, filed on Feb. 19, 1999, entitled "AN INTELLIGENT OBJECT-ORIENTED CONFIGURATION DATABASE SERIALIZER"; U.S. patent application No. 09/253,841, filed on Feb. 19, 1999, entitled "PROCESS FOR TRANSFORMING CONFIGURATION DATABASE GRAMMAR INTO INTERMEDIATE FORM THAT SIMPLIFIES DATABASE GENERATION"; U.S. patent application No. 09/253,866, filed on Feb. 19, 1999, entitled "GRAMMAR TO REPRESENT A HIERARCHICAL OBJECT-ORIENTED DATABASE"; U.S. patent application No. 09/253,867, filed on Feb. 19, 1999, entitled "TRANSFORMATION CUSTOMIZER FOR CONFIGURATION DATABASE COMPILATION AND SERIALIZATION PROCESSES"; U.S. patent application No. 09/079,501; U.S. patent application No. 09/079,042; U.S. patent application No. 09/079,500; U.S. patent application No. 09/079,103; U.S. patent application No. 09/079,102; U.S. patent application No. 09/079,499; U.S. patent application No. 09/079,043; and U.S. patent application No. 09/107,048, which are all incorporated herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/103; 707/100, 707/102, 707/104.1, 717/108, 717/116, 717/147, 717/153, 709/316, 709/332
US-CL-CURRENT: 707/100, 707/102, 707/104.1, 717/108, 717/116, 717/147, 717/153, 719/316, 719/332

FIELD-OF-SEARCH: 707/11, 707/100-104.1, 707/200, 709/201, 709/223, 709/230, 709/310, 709/316, 709/332, 717/1-11, 717/108, 717/116, 717/136, 717/140-148,

717/150-153, 717/118-119, 717/166-165

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5335346	August 1994	Fabbio	711/163
☐	5694598	December 1997	Durand et al.	707/103
☐	5758154	May 1998	Qureshi	713/1
☐	5797137	August 1998	Golshani et al.	707/4
☐	5822580	October 1998	Leung	707/103
☐	5893106	April 1999	Brobst et al.	707/102
☐	5899990	May 1999	Maritzen et al.	707/4
☐	6012067	January 2000	Sarkar	707/103
☐	6119129	September 2000	Traversat et al.	707/202
☐	6145121	November 2000	Levy et al.	703/22
☐	6199194	March 2001	Wang et al.	717/118
☐	6199195	March 2001	Goodwin et al.	717/1
☐	6298354	October 2001	Saulpaugh et al.	707/100

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 256 881	February 1988	EP	

OTHER PUBLICATIONS

Ege, R.K. et al., A modular Java API for object-oriented databases, computer software and applications conference, 1998, COMPSAC '98, Aug. 1998, pp. 55-60.*
Tennent, Principles of Programming Languages, Prentice Hall 1981, pp. 9-33.
Majka, "Getting Acquainted with NetInfo," NEXTSTEP In Focus, Summer 1993, vol. 3, Issue 3, copyright NeXT Computer, Inc. 1993. 10 pages.

ART-UNIT: 2177

PRIMARY-EXAMINER: Channavajjala; Srirama

ATTY-AGENT-FIRM: Meyertons Hood Kivlin Kowert & Goetzel, P.C. Kivlin; B. Noel

ABSTRACT:

A method and system for providing an intelligent intermediate form of an object-oriented database. The intermediate form is derived from a grammatical form of an

object-oriented database through the process of compilation. The grammatical form is a persistent form of an object-oriented database expressed in a human-readable and human-editable textual form according to a grammar. The intermediate form comprises an array of intelligent entry objects which encapsulate data with methods for manipulating that data. The methods include creating a database entry, creating a property associated with an entry, creating an attribute associated with an entry or property, querying the last entry, property, or attribute created, and finalizing entry storage. The intermediate form lacks the infrastructure of the database, but the intermediate form can be used to populate the object-oriented database with entries. The object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. The object-oriented database is platform-independent and is therefore configured to be hosted on several different operating systems and computing platforms.

33 Claims, 11 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L1: Entry 8 of 8 File: USPT Oct 2, 2001

US-PAT-NO: 6298354
DOCUMENT-IDENTIFIER: US 6298354 B1

TITLE: Mechanism and process to transform a grammar-derived intermediate form to an object-oriented configuration database

DATE-ISSUED: October 2, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Saulpaugh; Thomas E.	San Jose	CA		
Slaughter; Gregory L.	Palo Alto	CA		
Traversat; Bernard A.	San Francisco	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Sun Microsystems, Inc.	Palo Alto	CA			02

APPL-NO: 09/ 253839 [PALM]
DATE FILED: February 19, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS This application is related to U.S. patent application Ser. No. 09/253,840, filed on Feb. 19, 1999, entitled "AN INTELLIGENT OBJECT-ORIENTED CONFIGURATION DATABASE SERIALIZER"; U.S. patent application Ser. No. 09/253,841, filed on Feb. 19, 1999, entitled "PROCESS FOR TRANSFORMING CONFIGURATION DATABASE GRAMMAR INTO INTERMEDIATE FORM THAT SIMPLIFIES DATABASE GENERATION"; U.S. patent application Ser. No. 09/253,866, filed on Feb. 19, 1999, entitled "GRAMMAR TO REPRESENT A HIERARCHICAL OBJECT-ORIENTED DATABASF"; U.S. patent application Ser. No. 09/253,867, filed on Feb. 19, 1999, entitled "TRANSFORMATION CUSTOMIZER FOR CONFIGURATION DATABASE COMPILATION AND SERIALIZATION PROCESSES"; U.S. patent application Ser. No. 09/253,868, filed on Feb. 19, 1999, entitled "AN INTELLIGENT INTERMEDIATE STATE OF AN OBJECT-ORIENTED DATABASE"; U.S. patent application Ser. Nos. 09/079,501; 09/079,042; 09/079,500; 09/079,103; 09/079,102; 09/079,499; 09/079,043; and 09/107,048, which are all incorporated herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/103; 707/100
US-CL-CURRENT: 707/103R; 707/100

FIELD-OF-SEARCH: 707/103, 707/100

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5551029	August 1996	Jagadish et al.	707/103
☐	5694598	December 1997	Durand	707/103
☐	5758154	May 1998	Qureshi	713/1
☐	6182082	January 2001	Tanaka et al.	707/103

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 631 229	December 1994	EP	

OTHER PUBLICATIONS

Tennent, Principles of Programming Languages, Prentice Hall 1981, pp. 9-33.
Majka, "Getting Acquainted with NetInfo," Nextstep In Focus, Summer 1993, vol. 3, Issue 3, copyright NeXT Computer, Inc. 1993.

ART-UNIT: 217

PRIMARY-EXAMINER: Choules; Jack

ASSISTANT-EXAMINER: Lewis; Cheryl

ATTY-AGENT-FIRM: Conley, Rose & Tayon, PC Kivlin; B. Noel

ABSTRACT:

A method and system for transforming an intermediate form into an object-oriented database. The intermediate form is derived from a grammatical form of an object-oriented database through the process of compilation. The grammatical form is an expression of an object-oriented database in a textual form according to a grammar. The intermediate form comprises an array of intelligent entry objects that encapsulate data with methods for manipulating that data. The intermediate form comprises entries as in the object-oriented database but lacks the infrastructure of the database. The intermediate form can be used to populate the object-oriented database with entries. Population takes place through a public API for accessing the object-oriented database; in other words, through an interface which declares methods for navigating the database and adding entries to the database. The object-oriented database is an object-oriented configuration database which stores configuration parameters pertaining to the software and hardware of a computer system, such as application programs, device drivers, system services, and other components. The object-oriented database is platform-independent and is therefore configured to be hosted on several different operating systems and computing platforms.

20 Claims, 11 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L2: Entry 1 of 3 File: USPT Dec 30, 2003

DOCUMENT-IDENTIFIER: US 6671724 B1
TITLE: Software, systems and methods for managing a distributed network

Detailed Description Text (15):
The invented software, systems and methods may be configured using a third software component, to be later discussed in more detail with reference to FIGS. 5 and 13-16. Typically, this configuration utility is a platform-independent application that provides a graphical user interface for centrally managing configuration information for the control points and agents. In addition, the configuration utility may be adapted to communicate and interface with other management systems, including management platforms supplied by other vendors.

Detailed Description Text (84):
Referring now to FIGS. 13-16, both the agents and control points may be configured using configuration utility 106. Typically, configuration utility 106 is a platform-independent application that provides a graphical user interface for centrally managing configuration information for the control points and agents. To configure the control points and agents, the configuration utility interface with administrator module 134 of agent 70 and with administrator module 168 of control point 72. Alternatively, configuration utility 106 may interface with administrator module 168 of control point 72, and the control point in turn may interface with administrator module 134 of agent 70.

First Hit Fwd Refs

Generate Collection Print

L2: Entry 1 of 3 File: USPT Dec 30, 2003

US-PAT-NO: 6671724
DOCUMENT-IDENTIFIER: US 6671724 B1

TITLE: Software, systems and methods for managing a distributed network

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pandya; Suketu J.	Mission Viejo	CA		
Lakshmanan; Hariharan	Sunnyvale	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Centrisoft Corporation	Portland	OR			02

APPL-NO: 09/ 532101 [PALM]
DATE FILED: March 21, 2000

INT-CL: [07] G06 F 15/173

US-CL-ISSUED: 709/226; 709/224, 370/236
US-CL-CURRENT: 709/226; 370/236, 709/224

FIELD-OF-SEARCH: 709/223, 709/224, 709/225, 709/226, 709/232, 709/229, 709/234, 709/250, 370/395, 370/236, 710/17, 710/40, 714/20

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5461611	October 1995	Drake, Jr. et al.	
5517622	May 1996	Ivanoff et al.	
5548579	August 1996	Lebrun et al.	
5634006	May 1997	Baugher et al.	
5644715	July 1997	Baugher	
5655081	August 1997	Bonnell et al.	
5657452	August 1997	Kralowetz et al.	

☐	5689708	November 1997	Regnier et al.	709/229
☐	5694548	December 1997	Baugher et al.	
☐	5696486	December 1997	Poliquin et al.	
☐	5706437	January 1998	Kirchner et al.	
☐	5734865	March 1998	Yu	709/250
☐	5761428	June 1998	Sidey	
☐	5774656	June 1998	Hattori et al.	
☐	5777549	July 1998	Arrowsmith et al.	
☐	5781703	July 1998	Desai et al.	
☐	5793958	August 1998	Clement et al.	
☐	5794073	August 1998	Ramakrishnan et al.	710/40
☐	5799002	August 1998	Krishnan	370/234
☐	5822521	October 1998	Gartner et al.	
☐	5848266	December 1998	Scheurich	713/503
☐	5881313	March 1999	Ramakrishnan et al.	710/40
☐	5889958	March 1999	Willens	
☐	5892754	April 1999	Kompella et al.	
☐	5901142	May 1999	Averbuch et al.	
☐	5903568	May 1999	Tanaka et al.	
☐	5907324	May 1999	Larson et al.	
☐	5941947	August 1999	Brown et al.	
☐	5944783	August 1999	Nieten	
☐	5944795	August 1999	Civanlar	
☐	5948065	September 1999	Eilert et al.	709/226
☐	5953338	September 1999	Ma et al.	370/395
☐	5956482	September 1999	Agraharam et al.	
☐	5958010	September 1999	Agarwat et al.	709/224
☐	5968116	October 1999	Day, II et al.	
☐	5978594	November 1999	Bonnell et al.	710/17
☐	5983261	November 1999	Riddle	709/204
☐	5987611	November 1999	Freund	
☐	5996010	November 1999	Leong et al.	
☐	6018567	January 2000	Dulman	
☐	6085241	July 2000	Otis	709/223
☐	6097722	August 2000	Graham et al.	370/395
☐	6125390	September 2000	Touboul	709/223
☐	6199124	March 2001	Ramakrishnan et al.	710/40
	6279039	August 2001	Bhat et al.	709/226

☐				
☐	6400687	June 2002	Davison et al.	370/236
☐	6412000	June 2002	Riddle et al.	709/224
☐	6466978	October 2002	Mukherjee et al.	709/225
☐	6466980	October 2002	Lumnelsky et al.	709/226
☐	6502131	December 2002	Vaid et al.	709/224

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 95/27249	October 1995	WO	
WO 97/48214	December 1997	WO	
WO 98/26541	June 1998	WO	
WO 99/34544	July 1999	WO	

OTHER PUBLICATIONS

Moharpatra et al, ATM switching systems for broadband ISDN, Apr. 1997.*
Tsiotsios et al, ATM access-shaping and bandwidth-tuning of DQDB egress traffic, Sep. 1996.*
Orphanos et al., "An Integrated Application/Service Platform to Support Multimedia Applications," IEEE, pp. 1722-1726 (1994).
Lombardo et al., "A Model for Multimedia Service Creation and Activation," IEEE, pp. 1727-1733 (1994).
Nahrstedt et al., "The QoS Broker," IEEE, pp. 53-67 (1995).
Buddenberg, "Session Layer Requirements for Multicastinternets," IEEE, pp. 969-972 (1997).
Nishio et al., "Simplified Method for Session Coordination Using Multi-Level QoS Specification and Translation," IFIP, Ch. 38, pp. 333-344 (1997).
Youssef et al., "The Software Architecture of a Distributed Quality of Session Control Layer," IEEE, pp. 21-28 (1998).
Youssef et al., "Controlling Quality of Session in Adaptive Multimedia Multicast Systems," IEEE, pp. 160-167 (1998).
Youssef et al., "Configurable Multi-Agent System for QoS Control in WATM," IEEE, pp. 2882-2887 (1998).
Hong et al., "A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications," IEEE Network, pp. 70-79 (Mar./Apr. 1999).
Candan et al., "Least-Cost High-Quality Object Retrieval for Distributed Multimedia Collaborations," IEEE, pp. 649-654 (1999).
Cao et al., "An Architecture of Distributed Media Servers for Supporting Guaranteed QoS and Media Indexing," IEEE, pp. 1-5 (1999).
Cheong et al., "QoS Specification and Mapping for Distributed Multimedia Systems: A Survey of Issues," The Journal of Systems and Software, pp. 127-139 (1999).
Koga et al., "Interworking Architecture for Seamless Service Utilization between the TINA-Based Public Network and the Internet," 3529 SPIE 154-65 (1998).
Guedes et al., "An Agent-Based Approach for Supporting Quality of Service in Distributed Multimedia Systems," 21 Computer Communications 1269-78 (1998).
Ott et al., "An Architecture for Adaptive QoS and its Application to Multimedia Systems Design," 21 Computer Communications 334-49 (1998).
Li et al., "End-to-End QoS Support for Adaptive Applications Over the Internet," 3529 SPIE 166-76 (1998).
Alfano, "Design and Implementation of a Cooperative Multimedia Environment with QoS Control," 21 Computer Communications 350-61 (1998).

Almesberger et al., "Quality of Service Renegotiations," 3529 SPIE 124-31 (1998).

ART-UNIT: 2142

PRIMARY-EXAMINER: Powell; Mark R.

ASSISTANT-EXAMINER: Vu; Thong

ATTY-AGENT-FIRM: Kolisch Hartwell, P.C.

ABSTRACT:

Software, systems and methods for managing a distributed network environment including a plurality of computers interconnected by a network link, where at least some of the computers include a layered communications protocol stack for providing a data interface between an application program and the network link, the communications stack having a transport protocol layer for providing an end-to-end communications connection. The invention includes a control module and a plurality of agent modules, each agent being associated with one of the computers and adapted to dynamically monitor the associated computer at a data transmission point between an application program running on the computer and the transport protocol layer and repeatedly communicate with the control module in order to effect management of the distributed network system. The invented software, systems and methods may also include a messaging feature for providing users, IT personnel, or various management systems with informative messages concerning network conditions and network resources.

7 Claims, 19 Drawing figures

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 3 of 3 returned.

☐ 1. Document ID: US 20040030879 A1

Using default format because multiple data bases are involved.

L3: Entry 1 of 3 File: DWPI Feb 12, 2004

DERWENT-ACC-NO: 2004-190788
DERWENT-WEEK: 200418

TITLE: Generic provision of platform-specific configuration information to user of computer system, involves creating structures, associated with collected platform configuration information, in platform independent configuration layer tree

INVENTOR: JOHNSON, J G; ONUFER, G C ; SUBRAMANIAN, S ; ZATORSKI, R A

PRIORITY-DATA: 2002US-413016P (August 9, 2002), 2002US-0254952 (September 25, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20040030879 A1	February 12, 2004		012	G06F015/177

INT-CL (IPC): G06 F 15/177

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 2. Document ID: US 20030074512 A1

L3: Entry 2 of 3 File: DWPI Apr 17, 2003

DERWENT-ACC-NO: 2003-556004
DERWENT-WEEK: 200352

TITLE: Universal serial bus device configuration involves transmitting portion of configuration information including set and reset signals, from platform independent routine to universal serial bus through platform independent interface

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 3. Document ID: EP 663638 A1, DE 69428512 E, JP 07210488 A, US 5579529 A, EP 663638 B1

L3: Entry 3 of 3 File: DWPI Jul 19, 1995

DERWENT-ACC-NO: 1995-247663
DERWENT-WEEK: 200174

TITLE: User configuration for computer peripherals - has processor checking peripheral for configuration flag that if set causes it to look in extended ROM or disc for configuration file

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
(configur$5 near3 (information or data)) near10 (platform adj1 independent)	3

Display Format: - Change Format

Previous Page Next Page Go to Doc#

First Hit

☐ Generate Collection Print

L3: Entry 1 of 3 File: DWPI Feb 12, 2004

DERWENT-ACC-NO: 2004-190788
DERWENT-WEEK: 200418

TITLE: Generic provision of platform-specific configuration information to user of computer system, involves creating structures, associated with collected platform configuration information, in platform independent configuration layer tree

INVENTOR: JOHNSON, J G; ONUFER, G C ; SUBRAMANIAN, S ; ZATORSKI, R A

PATENT-ASSIGNEE: SUN MICROSYSTEMS INC (SUNM)

PRIORITY-DATA: 2002US-413016P (August 9, 2002), 2002US-0254952 (September 25, 2002)

Search Selected Search ALL Clear

PATENT-FAMILY:

	PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
☐	US 20040030879 A1	February 12, 2004		012	G06F015/177

APPLICATION-DATA:

PUB-NO	APPL-DATE	APPL-NO	DESCRIPTOR
US20040030879A1	August 9, 2002	2002US-413016P	Provisional
US20040030879A1	September 25, 2002	2002US-0254952	

INT-CL (IPC): G06 F 15/177

ABSTRACTED-PUB-NO: US20040030879A
BASIC-ABSTRACT:

NOVELTY - The method involves encapsulating a platform-specific code, associated with a hardware platform, into a plug-in module. Platform configuration information is collected. Structures, associated with the collected platform configuration information, are created in a platform independent configuration layer (PICL) tree. A user is provided with access to the PICL tree.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for a computer system.

USE - For generically providing platform-specific configuration information, associated with a hardware platform for a computer system, to a user of the computer system.

ADVANTAGE - Saves time and other resources in handling upgrades or changes to platforms.

DESCRIPTION OF DRAWING(S) - The figure shows the flow diagram of hardware platform

change handling method.

ABSTRACTED-PUB-NO: US20040030879A
EQUIVALENT-ABSTRACTS:

CHOSEN-DRAWING: Dwg.1/6

DERWENT-CLASS: T01
EPI-CODES: T01-F05B2;

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L1: Entry 1 of 1 File: USPT Dec 30, 2003

DOCUMENT-IDENTIFIER: US 6671724 B1
TITLE: Software, systems and methods for managing a distributed network

Detailed Description Text (15):
The invented software, systems and methods may be configured using a third software component, to be later discussed in more detail with reference to FIGS. 5 and 13-16. Typically, this configuration utility is a platform-independent application that provides a graphical user interface for centrally managing configuration information for the control points and agents. In addition, the configuration utility may be adapted to communicate and interface with other management systems, including management platforms supplied by other vendors.

Detailed Description Text (84):
Referring now to FIGS. 13-16, both the agents and control points may be configured using configuration utility 106. Typically, configuration utility 106 is a platform-independent application that provides a graphical user interface for centrally managing configuration information for the control points and agents. To configure the control points and agents, the configuration utility interface with administrator module 134 of agent 70 and with administrator module 168 of control point 72. Alternatively, configuration utility 106 may interface with administrator module 168 of control point 72, and the control point in turn may interface with administrator module 134 of agent 70.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L1: Entry 1 of 1 File: USPT Dec 30, 2003

US-PAT-NO: 6671724
DOCUMENT-IDENTIFIER: US 6671724 B1

TITLE: Software, systems and methods for managing a distributed network

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pandya; Suketu J.	Mission Viejo	CA		
Lakshmanan; Hariharan	Sunnyvale	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Centrisoft Corporation	Portland	OR			02

APPL-NO: 09/ 532101 [PALM]
DATE FILED: March 21, 2000

INT-CL: [07] G06 F 15/173

US-CL-ISSUED: 709/226; 709/224, 370/236
US-CL-CURRENT: 709/226; 370/236, 709/224

FIELD-OF-SEARCH: 709/223, 709/224, 709/225, 709/226, 709/232, 709/229, 709/234, 709/250, 370/395, 370/236, 710/17, 710/40, 714/20

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5461611	October 1995	Drake, Jr. et al.	
5517622	May 1996	Ivanoff et al.	
5548579	August 1996	Lebrun et al.	
5634006	May 1997	Baugher et al.	
5644715	July 1997	Baugher	
5655081	August 1997	Bonnell et al.	

☐	5657452	August 1997	Kralowetz et al.	
☐	5689708	November 1997	Regnier et al.	709/229
☐	5694548	December 1997	Baugher et al.	
☐	5696486	December 1997	Poliquin et al.	
☐	5706437	January 1998	Kirchner et al.	
☐	5734865	March 1998	Yu	709/250
☐	5761428	June 1998	Sidey	
☐	5774656	June 1998	Hattori et al.	
☐	5777549	July 1998	Arrowsmith et al.	
☐	5781703	July 1998	Desai et al.	
☐	5793958	August 1998	Clement et al.	
☐	5794073	August 1998	Ramakrishnan et al.	710/40
☐	5799002	August 1998	Krishnan	370/234
☐	5822521	October 1998	Gartner et al.	
☐	5848266	December 1998	Scheurich	713/503
☐	5881313	March 1999	Ramakrishnan et al.	710/40
☐	5889958	March 1999	Willens	
☐	5892754	April 1999	Kompella et al.	
☐	5901142	May 1999	Averbuch et al.	
☐	5903568	May 1999	Tanaka et al.	
☐	5907324	May 1999	Larson et al.	
☐	5941947	August 1999	Brown et al.	
☐	5944783	August 1999	Nieten	
☐	5944795	August 1999	Civanlar	
☐	5948065	September 1999	Eilert et al.	709/226
☐	5953338	September 1999	Ma et al.	370/395
☐	5956482	September 1999	Agraharam et al.	
☐	5958010	September 1999	Agarwat et al.	709/224
☐	5968116	October 1999	Day, II et al.	
☐	5978594	November 1999	Bonnell et al.	710/17
☐	5983261	November 1999	Riddle	709/204
☐	5987611	November 1999	Freund	
☐	5996010	November 1999	Leong et al.	
☐	6018567	January 2000	Dulman	
☐	6085241	July 2000	Otis	709/223
☐	6097722	August 2000	Graham et al.	370/395
☐	6125390	September 2000	Touboul	709/223
	6199124	March 2001	Ramakrishnan et al.	710/40

☐				
☐	6279039	August 2001	Bhat et al.	709/226
☐	6400687	June 2002	Davison et al.	370/236
☐	6412000	June 2002	Riddle et al.	709/224
☐	6466978	October 2002	Mukherjee et al.	709/225
☐	6466980	October 2002	Lumnelsky et al.	709/226
☐	6502131	December 2002	Vaid et al.	709/224

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 95/27249	October 1995	WO	
WO 97/48214	December 1997	WO	
WO 98/26541	June 1998	WO	
WO 99/34544	July 1999	WO	

OTHER PUBLICATIONS

Moharpatra et al, ATM switching systems for broadband ISDN, Apr. 1997.*
Tsiotsios et al, ATM access-shaping and bandwidth-tuning of DQDB egress traffic, Sep. 1996.*
Orphanos et al., "An Integrated Application/Service Platform to Support Multimedia Applications," IEEE, pp. 1722-1726 (1994).
Lombardo et al., "A Model for Multimedia Service Creation and Activation," IEEE, pp. 1727-1733 (1994).
Nahrstedt et al., "The QoS Broker," IEEE, pp. 53-67 (1995).
Buddenberg, "Session Layer Requirements for Multicastinternets," IEEE, pp. 969-972 (1997).
Nishio et al., "Simplified Method for Session Coordination Using Multi-Level QoS Specification and Translation," IFIP, Ch. 38, pp. 333-344 (1997).
Youssef et al., "The Software Architecture of a Distributed Quality of Session Control Layer," IEEE, pp. 21-28 (1998).
Youssef et al., "Controlling Quality of Session in Adaptive Multimedia Multicast Systems," IEEE, pp. 160-167 (1998).
Youssef et al., "Configurable Multi-Agent System for QoS Control in WATM," IEEE, pp. 2882-2887 (1998).
Hong et al., "A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications," IEEE Network, pp. 70-79 (Mar./Apr. 1999).
Candan et al., "Least-Cost High-Quality Object Retrieval for Distributed Multimedia Collaborations," IEEE, pp. 649-654 (1999).
Cao et al., "An Architecture of Distributed Media Servers for Supporting Guaranteed QoS and Media Indexing," IEEE, pp. 1-5 (1999).
Cheong et al., "QoS Specification and Mapping for Distributed Multimedia Systems: A Survey of Issues," The Journal of Systems and Software, pp. 127-139 (1999).
Koga et al., "Interworking Architecture for Seamless Service Utilization between the TINA-Based Public Network and the Internet," 3529 SPIE 154-65 (1998).
Guedes et al., "An Agent-Based Approach for Supporting Quality of Service in Distributed Multimedia Systems," 21 Computer Communications 1269-78 (1998).
Ott et al., "An Architecture for Adaptive QoS and its Application to Multimedia Systems Design," 21 Computer Communications 334-49 (1998).
Li et al., "End-to-End QoS Support for Adaptive Applications Over the Internet," 3529 SPIE 166-76 (1998).

Alfano, "Design and Implementation of a Cooperative Multimedia Environment with QoS Control," 21 Computer Communications 350-61 (1998).
Almesberger et al., "Quality of Service Renegotiations," 3529 SPIE 124-31 (1998).

ART-UNIT: 2142

PRIMARY-EXAMINER: Powell; Mark R.

ASSISTANT-EXAMINER: Vu; Thong

ATTY-AGENT-FIRM: Kolisch Hartwell, P.C.

ABSTRACT:

Software, systems and methods for managing a distributed network environment including a plurality of computers interconnected by a network link, where at least some of the computers include a layered communications protocol stack for providing a data interface between an application program and the network link, the communications stack having a transport protocol layer for providing an end-to-end communications connection. The invention includes a control module and a plurality of agent modules, each agent being associated with one of the computers and adapted to dynamically monitor the associated computer at a data transmission point between an application program running on the computer and the transport protocol layer and repeatedly communicate with the control module in order to effect management of the distributed network system. The invented software, systems and methods may also include a messaging feature for providing users, IT personnel, or various management systems with informative messages concerning network conditions and network resources.

7 Claims, 19 Drawing figures

First Hit

End of Result Set

Generate Collection Print

L3: Entry 3 of 3 File: DWPI Jul 19, 1995

DERWENT-ACC-NO: 1995-247663
DERWENT-WEEK: 200174

TITLE: User configuration for computer peripherals - has processor checking peripheral for configuration flag that if set causes it to look in extended ROM or disc for configuration file

INVENTOR: AMSDEN, J D; BURKE, T C ; KAISNER, J W ; TERRELL, M R ; TODD, D K

PATENT-ASSIGNEE: NCR INT INC (NATC), AT & T GLOBAL INFORMATION SOLUTIONS INT (AMTT), NCR CORP (NATC)

PRIORITY-DATA: 1993US-0176022 (December 30, 1993), 1995US-0424227 (April 19, 1995)

Search Selected Search ALL Clear

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
EP 663638 A1	July 19, 1995	E	017	G06F013/38
DE 69428512 E	November 8, 2001		000	G06F013/38
JP 07210488 A	August 11, 1995		014	G06F013/10
US 5579529 A	November 26, 1996		017	G06F013/00
EP 663638 B1	October 4, 2001	E	000	G06F013/38

DESIGNATED-STATES: DE FR GB DE FR GB

CITED-DOCUMENTS:EP 192066; EP 48781 ; US 5237689

APPLICATION-DATA:

PUB-NO	APPL-DATE	APPL-NO	DESCRIPTOR
EP 663638A1	December 21, 1994	1994EP-0309649	
DE 69428512E	December 21, 1994	1994DE-0628512	
DE 69428512E	December 21, 1994	1994EP-0309649	
DE 69428512E		EP 663638	Based on
JP 07210488A	December 26, 1994	1994JP-0336729	
US 5579529A	December 30, 1993	1993US-0176022	Cont of
US 5579529A	April 19, 1995	1995US-0424227	
EP 663638B1	December 21, 1994	1994EP-0309649	

INT-CL (IPC): G06 F 13/00; G06 F 13/10; G06 F 13/38; G06 F 15/16

ABSTRACTED-PUB-NO: EP 663638A
BASIC-ABSTRACT:

The computer system includes a standard for configuring computer peripherals connected to a PCI bus. The computer has a processor system (12) with a PCI bus (10). A number of peripherals are connected to this bus such as audio (14), video (16) or graphics (18). These require to be configured when installed.

The processor check a flag in each of the peripherals. If the flag is in one condition no special action is required. If the flag is in the 'user specification mode' then extended ROM or a disc is checked for a configuration file. This allows a sequence of setting to be determined from the user.

ADVANTAGE - Provides a standard format for configuration data that is platform independent.

ABSTRACTED-PUB-NO: EP 663638B
EQUIVALENT-ABSTRACTS:

The computer system includes a standard for configuring computer peripherals connected to a PCI bus. The computer has a processor system (12) with a PCI bus (10). A number of peripherals are connected to this bus such as audio (14), video (16) or graphics (18). These require to be configured when installed.

The processor check a flag in each of the peripherals. If the flag is in one condition no special action is required. If the flag is in the 'user specification mode' then extended ROM or a disc is checked for a configuration file. This allows a sequence of setting to be determined from the user.

ADVANTAGE - Provides a standard format for configuration data that is platform independent.

US 5579529A

A method for configuring a peripheral coupled to a computer, the peripheral comprising a memory including an indicating register signifying whether the peripheral has user-selectable operating parameters and a configuration space, the configuration space comprising at least one configuration file and one or more configuration registers for setting the user-selectable operating parameters of the peripheral, each configuration file having a version indicator and a plurality of user selectable settings, the method comprising the steps of:

(a) accessing the indicating register in the peripheral;

(b) accessing the configuration space in the peripheral when the value stored in the indicating register indicates that the peripheral has one or more user-selectable operating parameters;

(c) interpreting the configuration file's contents to identify the user selectable settings for the user-selectable operating parameters of the peripheral and to determine how the user-selectable setting options should be presented to the user, comprising the steps of reading the version indicator and identifying the user-selectable settings according to the version indicator;

(d) presenting the user-selectable settings for the user-selectable operating

parameters of the peripheral to a user;

(e) accepting a selection of one of the user-selectable settings for each of the user-selectable operating parameters of the peripheral from the user in the computer; and

(f) setting the operating parameters of the peripheral according to the selected settings by storing a value from the configuration file into one or more of the configuration registers in the configuration space.

CHOSEN-DRAWING: Dwg.1/5 Dwg.5a/5

DERWENT-CLASS: T01
EPI-CODES: T01-C07D; T01-H05A;

US-PAT-NO: 6450885

DOCUMENT-IDENTIFIER: US 6450885 B2

TITLE: Method and apparatus for securing electronic games

---------- KWIC ---------

Detailed Description Text - DETX (23):

Referring again to FIG. 2, network interface 245 is the gateway to communicate with players through respective player terminals 300. Conventional internal or external modems may serve as network interface 245. Network interface 245 preferably supports modems at a range of baud rates from 1200 upward, but may combine such inputs into a T1 or T3 line if more bandwidth is required. In a preferred embodiment, network interface 245 is connected to the Internet and/or a commercial on-line service such as America Online, CompuServe, or Prodigy, allowing players access to game server 200 from a wide range of electronic connections. Several commercial electronic mail servers include the above functionality. For example, NCD Software manufactures "Post.Office," a secure server-based electronic mail software package designed to link people and information over enterprise networks and the Internet. This product is platform independent and utilizes open standards based on Internet protocols. Users can exchange messages with enclosures such as files, graphics, video, and audio. This product also supports multiple languages. Alternatively, network interface 245 may be configured as a voice-mail interface, web site, electronic Bulletin Board System ("BBS"), or electronic-mail address.

First Hit Fwd Refs

Generate Collection Print

L3: Entry 3 of 4 File: USPT Mar 19, 2002

US-PAT-NO: 6359976
DOCUMENT-IDENTIFIER: US 6359976 B1

TITLE: System and method for monitoring service quality in a communications network

DATE-ISSUED: March 19, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kalyanpur; Gaurang S.	Allen	TX		
Harper; Chad Daniel	McKinney	TX		
Brehm; Grant Michael	McKinney	TX		
Chan; Chunchun Jonina	Plano	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Inet Technologies, Inc.	Richardson	TX			02

APPL-NO: 09/ 395801 [PALM]
DATE FILED: September 14, 1999

PARENT-CASE:
RELATED APPLICATIONS This application is a continuation-in-part of pending application Ser. No. 09/093,955, filed on Jun. 8, 1998, entitled SYSTEM AND METHOD FOR MONITORING SERVICE QUALITY IN A COMMUNICATIONS NETWORK; and Ser. No. 09/093,824, now U.S. Pat. No. 6,249,675, entitled TRANSACTION CONTROL APPLICATION PART (TCAP) CALL DETAIL RECORD GENERATION IN A COMMUNICATIONS NETWORK, filed Jun. 8, 1998 and issued Jun. 19, 2001. The disclosure of which are hereby incorporated by reference herein. The present invention is also related to the following pending applications: Ser. No. 09/057,940, entitled SYSTEM AND METHOD FOR MONITORING PERFORMANCE STATISTICS IN A COMMUNICATIONS NETWORK, filed Apr. 9, 1998; Ser. No. 09/092,428, entitled SYSTEM AND METHOD FOR DETECTING HIGH MESSAGE TRAFFIC LEVELS IN A COMMUNICATIONS NETWORK; Ser. No. 09/092,699, entitled SYSTEM AND METHOD FOR SIGNAL UNIT DATA STORAGE AND POST CAPTURE CALL TRACE IN A COMMUNICATIONS NETWORK; Ser. No. 09/092,256, entitled SYSTEM AND METHOD FOR GENERATING QUALITY OF SERVICE STATISTICS FOR AN INTERNATIONAL COMMUNICATIONS NETWORK; and Ser. No. 09/092,771, entitled SYSTEM AND METHOD FOR CORRELATING TRANSACTION MESSAGES IN A COMMUNICATIONS NETWORK, all filed Jun. 5, 1998; and Ser. No. 09/094,122, entitled SYSTEM AND METHOD FOR CORRELATING TRANSACTION MESSAGES IN A COMMUNICATIONS NETWORK, filed Jun. 9, 1998; and Ser. No. 09/156,328, entitled SYSTEM AND METHOD FOR MONITORING LINK STATUS IN A COMMUNICATION NETWORK, filed Sep. 18, 1998. These applications are commonly assigned and are hereby incorporated by reference herein.

INT-CL: [07] H04 M 1/24, H04 M 15/00

US-CL-ISSUED: 379/134; 379/32.01, 379/32.02, 379/133, 379/111, 379/112.01, 379/114.01
US-CL-CURRENT: 379/134; 379/111, 379/112.01, 379/114.01, 379/133, 379/32.01,

379/32.02

FIELD-OF-SEARCH: 379/34, 379/112, 379/113, 379/114, 379/133, 379/134, 379/139, 379/140, 379/210, 379/32.01, 379/32.02, 379/111, 379/112.01, 379/112.06, 379/112.07

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5008929	April 1991	Olsen et al.	379/112
5333183	July 1994	Herbert	379/112
5426688	June 1995	Anand	379/5
5438570	August 1995	Karras et al.	370/94.2
5448624	September 1995	Hardy et al.	379/67
5457729	October 1995	Hamann et al.	379/2
5473596	December 1995	Garafola et al.	370/13
5488648	January 1996	Womble	375/13
5521902	May 1996	Ferguson	370/13
5539804	July 1996	Hong et al.	379/33
5550914	August 1996	Clarke et al.	379/230
5550984	August 1996	Gelb	395/200.17
5579371	November 1996	Aridas et al.	379/34
5590171	December 1996	Howe et al.	379/33
5592530	January 1997	Brockman et al.	379/34
5675635	October 1997	Vos et al.	379/113
5680437	October 1997	Segal	379/10
5680442	October 1997	Bartholomew et al.	379/67
5694451	December 1997	Arinell	379/34
5699348	December 1997	Baidon et al.	370/242
5699412	December 1997	Polcyn	379/89
5703939	December 1997	Bushnell	379/113
5706286	January 1998	Reiman et al.	370/401
5712908	January 1998	Brinkman et al.	379/119
5729597	March 1998	Bhusri	379/115
5737399	April 1998	Witzman et al.	379/112
5757895	May 1998	Aridas et al.	379/136
5793771	August 1998	Darland et al.	370/467

☐	5799073	August 1998	Fleischer, III et al.	379/113
☐	5822401	October 1998	Cave et al.	379/34
☐	5825769	October 1998	O'Reilly et al.	370/360
☐	5828729	October 1998	Clermont et al.	379/34
☐	5854824	December 1998	Bengal et al.	379/34
☐	5854835	December 1998	Montgomery et al.	379/119
☐	5867558	February 1999	Swanson	379/34
☐	5875238	February 1999	Glitho et al.	375/116
☐	5881132	March 1999	O'Brien et al.	379/35
☐	5883948	March 1999	Dunn	379/210
☐	5892812	April 1999	Pester, III	379/34
☐	5912954	June 1999	Whited et al.	379/115
☐	5920613	July 1999	Alcott et al.	379/114
☐	5999604	December 1999	Walter	379/133
☐	6028914	February 2000	Lin et al.	379/14

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0541145	October 1992	EP	
WO 97/05749	February 1970	WO	
WO 95/33352	December 1995	WO	
WO 97/05749	February 1997	WO	
WO98/47275	October 1998	WO	

OTHER PUBLICATIONS

International Search Report (PCT/US 00/25070) dated Dec. 14, 2000.
George Pavlou et al., Intelligent Remote Monitoring, Oct. 16, 1995.

ART-UNIT: 2643

PRIMARY-EXAMINER: Tieu; Binh

ASSISTANT-EXAMINER: Tran; Quoc D.

ATTY-AGENT-FIRM: Fulbright & Jaworski LLP

ABSTRACT:

A system and method for monitoring service quality using Call Detail Records (CDR) in a communications network, such as a Signaling System No. 7 (SS7) network, is disclosed. Network monitors capture substantially all signaling units in the SS7 network generate a complete record for all calls, transactions and other communications over the network. Users configure CDR profiles that are used to filter the records. A CDR application filters the records by parsing out signaling unit components that have been selected by the user in the CDR profile. The

selected message components are then formatted into a CDR record, which is sent to an external system that generates certain statistics for the message records and stores the statistics to a database. A report application recalls the statistics from the database and presents statistics in a reporting format configured by the user. The reports indicate the statistical performance of network providers for selected called or calling telephone numbers or for selected services. The CDRs and statistics are available to a user either in real-time or in response to a query of historical CDR data. The network quality monitoring system is separate and independent from the network monitoring equipment.

10 Claims, 5 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L3: Entry 4 of 4 File: USPT Nov 17, 1998

DOCUMENT-IDENTIFIER: US 5838165 A
TITLE: High performance self modifying on-the-fly alterable logic FPGA, architecture and method

Detailed Description Text (3):
As before mentioned, FIG. 1 shows the prior art FPGA interface using FPGA units (#1-#n) programmed through a bus system, so-labelled, by a CPU working with main memory. The FPGA units have the internal logic cell structure or array and routing channels of FIG. 2, with each configurable logic cell having logic functions controlled by an associated configuration static RAM (SRAM) as represented in FIG. 3, wherein the configuration data is stored in the small localized internal static RAM bits. In this invention, however, a DRAM core, as shown in FIG. 4, is used to store multiple configurations. The configuration SRAM bits required for the device configuration are also provided. A DRAM row wide bus is provided (where the term "row" as used herein also embraces a part or fraction of a row), which connects directly to the SRAM bits, which then subsequently control the programmable elements. After a configuration command is given, a row is retrieved each time and is stored in the said SRAM bits, until all the necessary configuration storage elements have been loaded. A partial reconfiguration is achieved by only loading those SRAM bits which need be changed. A further enhancement of this invention is to provide masking capability such that only bits which need be changed are allowed to be loaded into configuration SRAM. It is also possible to load new configuration data into the DRAM while the chip is operational. The same DRAM is also usable as storage space accessible from both the external I/O or via the internal logic. From the external interface side, it will have a narrow I/O width data interface, but internally its row wide bus can be used to store/retrieve maximum of a row wide data in one access. Once a row has been selected internally, the capability to access the column data at very high speed makes this an ideal space for state machine usage. It is not necessary to follow the traditional, equal number of rows and columns approach; and in some cases, it may be advantageous to have a structure with considerable bias towards rows as distinguished from columns, to allow for even faster dynamic reconfiguration. The over all functionality can be best illustrated with an example.

Detailed Description Text (12):
Consider, for example, `JAVA` (the most widely used Internet language) which is specifically designed to be platform independent, thereby providing complete portability among various machines. The disadvantage is that `JAVA` runs extremely slow as it does not take advantage of unique architectural capabilities of different CPUs. One approach to improve `JAVA` execution speed is by providing it a common virtual hardware platform (in addition to the traditional CPU functionality), alterable at very high speed. This virtual hardware is implemented either by providing reconfigurability with `SONAL` capability on CPUs themselves or implemented as a separate `SONAL` FPGA. This architecture thus retains the key element of platform independence, yet provides higher speed execution.

CLAIMS:

4. A method of configuring an array of programmable logic cells each having logic functions controlled by an associated configuration bit memory, that comprises, storing bit information defining multiple program configurations in a DRAM core; connecting a bus to the array to enable a DRAM row wide loading of the configuration bit memories of the cells with bit information defining a desired configuration; and, upon a configuration command, retrieving from the DRAM core, at least a row at a time, the configuration bit information, and loading such information in the bit memories of the cells to control the corresponding cell logic functions to achieve the desired configuration programming, and in which the DRAM core is also used as storage space accessible from both external I/O interfacing and internal logic.

22. Apparatus for configuring an array of programmable logic cells each having logic functions controlled by an associated configuration bit memory, the apparatus having, in combination, a DRAM core for storing bit information defining multiple program configurations; a bus interconnecting the array and the DRAM core such as to enable a DRAM row wide loading of the configuration bit memories of the array of logic cells with bit information defining a desired configuration; means for generating a configuration command; and, upon such configuration command, and responsive thereto, means for retrieving from the DRAM core, at least a row at a time, the configuration bit information; and means for loading such information in said bit memories of the logic cells to control the corresponding cell logic functions to achieve the desired configuration programming, and in which the DRAM core is also used as storage space accessible from both external I/O interfacing and internal logic.

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L3: Entry 4 of 4 File: USPT Nov 17, 1998

US-PAT-NO: 5838165
DOCUMENT-IDENTIFIER: US 5838165 A

TITLE: High performance self modifying on-the-fly alterable logic FPGA, architecture and method

DATE-ISSUED: November 17, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chatter; Mukesh	Hopkinton	MA	01748	

APPL-NO: 08/ 700966 [PALM]
DATE FILED: August 21, 1996

INT-CL: [06] H03 K 19/173, H03 K 7/38

US-CL-ISSUED: 326/38; 326/40, 326/39
US-CL-CURRENT: 326/38; 326/39, 326/40

FIELD-OF-SEARCH: 365/236, 326/38-41

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4831573	May 1989	Norman	326/39
4940909	July 1990	Mulder et al.	326/38
5375086	December 1994	Wahlstrom	326/40
5426378	June 1995	Ong	326/38
5469003	November 1995	Kean	326/38
5488582	January 1996	Camarota	365/51
5581198	December 1996	Trimberger	326/38
5646544	July 1997	Iadanza	326/38

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
62-269420	November 1987	JP	326/38
62-269422	November 1987	JP	326/38

ART-UNIT: 289

PRIMARY-EXAMINER: Tokar; Michael J.

ASSISTANT-EXAMINER: Roseen; Richard

ATTY-AGENT-FIRM: Rines and Rines

ABSTRACT:

A technique for configuring arrays of programmable logic cells, including those associated with FPGA devices, through a novel DRAM-based configuration control structure that enables not only "on-the-fly" alterable chip and similar device reconfigurations, but, where desired, self-modifying reconfigurations for differing functionalities of the devices, eliminating current serious reconfigurability limitations and related problems, while providing significantly enhanced system performance at low cost. A large amount of memory is available internal to the FPGA and is accessed with a small number of pins such that the reconfiguration time is, for example, four orders of magnitude faster than the traditional approaches and at notably low cost.

24 Claims, 8 Drawing figures

Refine Search

Search Results -

Terms	Documents
L5 and (platform adj1 independent)	62

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, May 26, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L6	L5 and (platform adj1 independent)	62	L6
L5	L1 and ((external or outside) adj1 (device or module or peripheral or (I adj1 O)))	5946	L5
	DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L4	L3	0	L4
	DB=USPT; PLUR=YES; OP=OR		
L3	L2 and (platform adj1 independent)	4	L3
L2	L1 same ((external or outside) adj1 (device or module or peripheral or (I adj1 O)))	736	L2
L1	configur$5 near5 (data or information)	62677	L1

END OF SEARCH HISTORY

Your search matched **11** of **1040503** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or enteri new one in the text box.

config* and (information or data)<and>(platform inde Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Integrating information appliances into an interactive workspace**
Fox, A.; Johanson, B.; Hanrahan, P.; Winograd, T.;
Computer Graphics and Applications, IEEE , Volume: 20 , Issue: 3 , May-June

Pages:54 - 65

[Abstract] [PDF Full-Text (408 KB)] **IEEE JNL**

---

2 **Utilizing abstract Web engineering concepts: an architecture**
Heberle, A.; Rehse, J.; Onasch, B.; Sieling, B.;
System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on , 3-6 Jan. 2001
Pages:8 pp.

[Abstract] [PDF Full-Text (732 KB)] **IEEE CNF**

---

3 **State estimation of electric power system for new technological sys**
Grishin, Yu.A.; Kolosok, I.N.; Korkina, E.S.; Em, L.V.;
Electric Power Engineering, 1999. PowerTech Budapest 99. International Conference on , 29 Aug.-2 Sept. 1999
Pages:273

[Abstract] [PDF Full-Text (100 KB)] **IEEE CNF**

---

4 **IVORY-an object-oriented framework for physics-based informatior visualization in Java**
Sprenger, T.C.; Gross, M.H.; Bielser, D.; Strasser, T.;
Information Visualization, 1998. Proceedings. IEEE Symposium on , 19-20 Oc 1998
Pages:79 - 86, 155

Bickley, M.; Kewisch, J.;
Particle Accelerator Conference, 1993., Proceedings of the 1993 , 17-20 May
Pages:1835 - 1837 vol.3

[Abstract] [PDF Full-Text (304 KB)] **IEEE CNF**
